Desenvolupament d'una aplicació progressiva de *carpooling* laboral

Salvador Navarrete Garcia
Grau d'Enginyeria Informàtica
Desenvolupament d'aplicacions en dispositius mòbils (HTML5 o Windows Phone)

Carles Garrigues Olivella
Jordi Almirall López
Carles Sànchez Rosa

5 de juny de 2020

FITXA DEL TREBALL FINAL

Títol del treball:	*Desenvolupament d'una aplicació progressiva de carpooling laboral*
Nom de l'autor:	*Salvador Navarrete Garcia*
Nom del consultor/a:	*Jordi Almirall López* *Carles Sànchez Rosa*
Nom del PRA:	*Carles Garrigues Olivella*
Data de lliurament (mm/aaaa):	*06/2020*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*Desenvolupament d'aplicacions en dispositius mòbils (HTML5 o Windows Phone)*
Idioma del treball:	*Català*
Paraules clau	*mobilitat, carpooling, pwa*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

Aquest treball adreça el problema de la mobilitat interurbana, específicament els desplaçaments relacionats amb la presència laboral, allò que es denomina *commuting* en anglès, és a dir, l'anada i tornada a la feina. La proposta de l'autor és una aplicació que permeti als conductors habituals compartir les places lliures dels seus vehicles a canvi de compartir costos.

Per al disseny de l'aplicació s'han analitzat en detall els usuaris potencials, les seves necessitats i els contexts d'ús, informació de la qual s'ha obtingut els requisits funcionals. S'han realitzat prototips, primer a mà alçada, i després d'alta fidelitat; i posteriorment s'han avaluat i descrit alguns dels aspectes únics que introdueix l'aplicació.

Per a la implementació s'han utilitzat tecnologies web d'última generació, conjuntament amb les noves eines orientades a desenvolupar aplicacions progressives, com els *service workers* o les *push notifications*.

El resultat final és una aplicació funcional, la qual implementa totes les funcionalitats projectades en l'apartat de planificació.

Abstract (in English, 250 words or less):

This thesis looks into the problem of interurban mobility specifically for work commuting. The author proposes a mobile application that allows drivers to share the free seats in their cars as well as the overall cost with other users.

For the design of the application, potential users, their needs and its use cases have been analyzed, and from these insights, functional requirements have been developed. Both drawings and high-fidelity prototypes have been created and evaluated, while specific, new features from this project are explained.

For the implementation, cutting edge web technologies have been used, together with new tools created for the development of progressive web applications, such as service workers and push notifications.

The end result is a functional application which implements all the projected features in the planning section.

Índex

1. Introducció

1.1 Context i justificació del Treball

Aquest treball adreça el problema de la mobilitat interurbana, específicament els desplaçaments relacionats amb la presència laboral, allò que es denomina *commuting* en anglès, és a dir, l'anada i tornada a la feina. Tot i que es tracta d'un fenomen global, el projecte se centra en la ciutat de Barcelona i la seva àrea metropolitana.

A continuació s'examinen les problemàtiques existents:

- **Contaminació i escalfament global**: tot i que encara no hi ha un consens global sobre els efectes de la contaminació envers l'escalfament global, és innegable que la temperatura del planeta està en augment i que la contaminació és un factor que hi contribueix significativament. Es calcula que el transport genera al voltant del 30% de la contaminació de gasos d'efecte hivernacle[1]. En els últims anys s'està començant a apreciar l'impacte de l'escalfament global amb l'augment en la freqüència d'episodis climàtics anòmals i/o extrems arreu del món. Aquest fenomen afecta no només la viabilitat del model productiu actual, sinó que amenaça la mateixa habitabilitat del planeta, per la qual cosa, la seva justificació hauria de ser evident.
- **Saturació de les vies de comunicació:** L'activitat econòmica es concentra cada vegada més en les grans ciutats, especialment les noves indústries i llocs de treball del futur, ja que és en aquests grans nuclis urbans on obtenen accés a tota mena de serveis i talent. Alhora, el cost de la vida a les grans capitals s'ha encarit molt durant els últims anys a causa de l'increment del preu de l'habitatge[2], i això està provocant que població local estigui traslladant-se a ciutats properes més assequibles[3]. En definitiva, aquests dos efectes, entre d'altres, fa que la mobilitat interurbana, és a dir, de ciutats de l'àrea metropolitana cap a la capital catalana, augmenti. Es calcula que més de 150.000 vehicles privats accedeixen a la ciutat de Barcelona cada dia i que aquest fenomen podria tenir un impacte de 137 milions d'euros[4].
- **Zones de baixes emissions (ZBE)**: la proposta de les administracions de les grans capitals ha estat la prohibició de l'entrada als vehicles més contaminants dins determinades zones de la ciutat. D'aquesta manera es dificulta l'accés i es fa necessària una renovació del parc automobilístic enmig d'una etapa d'alts i baixos econòmics.

Per tal de fer front a aquestes problemàtiques es proposa crear l'aplicació Ronda amb l'objectiu d'incrementar l'ocupació del vehicle privat en els trajectes interurbans per transportar-se al lloc de treball. Actualment l'ocupació dins aquest tipus de trajectes acostuma a ser d'una persona, la qual cosa representa una enorme ineficiència (en la majoria dels casos una persona utilitza l'energia que podria utilitzar-se per transportar fins a 5 persones).

1.2 Objectius del Treball

Objectius generals

- Disseny d'un producte i una experiència d'usuari innovadors i diferenciats els quals permetin i incentivin el *car pooling* entre usuaris de vehicle privat.
- Desenvolupament d'una aplicació web multiplataforma a través del paradigma d'aplicacions web progressives (PWA).
- Desenvolupament d'una API REST que funcioni com a *backend* de l'aplicació.
- Demostrar l'estat de l'art de les tecnologies web en relació amb el desenvolupament d'aplicacions per dispositius mòbils.

Objectius funcionals de l'aplicació

Tot usuari podrà:

- Registrar-se i identificar-se.
- Registrar-se com a conductor.
- Definir els trajectes d'anada i tornada del seu lloc de treball.
- Trobar trajectes d'altres conductors i sol·licitar places dins dels seus vehicles.
- Realitzar pagaments als conductors pels trajectes realitzats.
- Gestionar les sol·licituds i trajectes confirmats (cancel·lacions, etc.).
- Personalitzar el seu perfil públic.
- Descobrir nous usuaris i guardar-los com a contactes.

Tot conductor podrà:

- Establir la seva visibilitat (pública o privada).
- Acceptar o rebutjar sol·licituds.
- Cancel·lar trajectes reservats.
- Gestionar la seva disponibilitat.
- Modificar els trajectes en qualsevol moment (imprevistos, etc.)

Els objectius subratllats es consideren com a crítics per l'èxit de projecte.

1.3 Enfocament i mètode seguit

L'estratègia triada és la de crear un nou producte. Un dels elements fonamentals d'aquest projecte és l'experiència d'usuari i les funcionalitats específiques fortament enfocades als desplaçaments diaris per traslladar-se al lloc de treball, per tant no és possible adaptar una aplicació o solució existents. Aquests tipus de desplaçaments comporten una sèrie de casuístiques les quals són úniques:

- La necessitat de comunicació prèvia entre conductor i passatger(s) per tal d'establir un punt de trobada.
- El fet que són viatges recurrents (tant conductor com usuaris tenen un origen i destí, i un horari fix).
- La flexibilitat horària (en el dia a dia sorgeixen imprevistos: reunions que s'allarguen, despertadors que no sonen…).

S'ha de remarcar que l'usuari no obtindrà un benefici econòmic significatiu que pugui justificar un gran esforç per la seva part, per tant, crec que l'èxit d'aquest projecte estarà molt relacionat amb la facilitat d'ús i la capacitat del conductor en compartir el seu vehicle amb el mínim esforç possible per tal que l'incentiu obtingut resulti atractiu.

Atès que el projecte es realitza de manera individual, i s'ha d'acotar i planificar de manera que es compleixi el calendari d'avaluació, la metodologia més adequada és la de **desenvolupament en cascada**. Aquesta metodologia prioritza la documentació i la planificació envers la col·laboració i la iteració. Per tant, s'estructura el projecte complet i s'executa seguint aquesta especificació fins a la seva finalització, el qual és exactament el que es duu a terme amb aquest treball.

1.4 Planificació del Treball

S'estipula com a disponibilitat horària:

- 4 hores en dies laborables
- 6 hores en dies festius

Es pren com una estimació i hi haurà flexibilitat en funció de la càrrega de treball.

La planificació s'estructura al voltant de les proves d'avaluació continuada (PACs), les quals defineixen les entregues parcials a realitzar:

Fita	Data inici	Data final	Durada (hores)
PAC1: Pla de treball	19/02/2020	04/03/2020	20
Documentar context, objectius i planificació	19/02/2020	01/03/2020	10
Desenvolupar aplicació “Hello world”	02/03/2020	04/03/2020	10
PAC2: Disseny i arquitectura	05/03/2020	01/04/2020	70
Definició d’usuaris i context d’ús	05/03/2020	06/03/2020	6
Disseny conceptual	07/03/2020	09/03/2020	12
Prototipatge	10/03/2020	14/03/2020	20
Avaluació	15/03/2020	16/03/2020	6
Definició casos d’ús	17/03/2020	22/03/2020	10
Disseny de l’arquitectura	23/03/2020	29/03/2020	10
Revisió i correccions	30/03/2020	01/04/2020	6
PAC3: Implementació	02/04/2020	13/05/2020	195
Implementació de la interfície d’usuari (html5, css, vue)	02/04/2020	11/04/2020	40
Desenvolupament de la lògica de client (vue, vuex)	12/04/2020	20/04/2020	40
Configuració de base de dades (MongoDB)	21/04/2020	21/04/2020	4
Desenvolupament API servidor (Node.js)	22/04/2020	01/05/2020	40
Integració de servei d’autenticació (Firebase)	02/05/2020	07/05/2020	25
Desenvolupament del *Service Worker* (PWA)	08/05/2020	08/05/2020	4
Desenvolupament i execució de	09/05/2020	13/05/2020	20

proves (jest)			
Lliurament final	14/05/2020	05/06/2020	65
Preparació lliurament de codi	14/05/2020	17/05/2020	10
Finalització de la memòria	18/05/2020	21/05/2020	10
Presentació en vídeo	22/05/2020	03/06/2020	40
Redacció autoinforme	04/06/2020	05/06/2020	5

1.5 Breu sumari de productes obtinguts

- Codi font de l'aplicació (.zip)
- Memòria del treball
- Presentació en vídeo

1.6 Breu descripció dels altres capítols de la memòria

Capítol 2: Anàlisi

- Revisió dels productes i solucions existents al mercat que es podrien considerar com a competidors o substitutius.
- Públic objectiu i perfil d'usuari: descripció de les tipologies d'usuaris a les quals s'adreça el producte.
- Proposta de valor: anàlisi dels incentius i obstacles per cada tipologia d'usuari.

Capítol 3: Disseny i experiència d'usuari

- Experiència d'usuari: descripció i argumentació dels principals elements que conformen l'experiència d'usuari tals com l'estructura de l'aplicació i els fluxos d'interacció.
- Disseny gràfic i interfícies: descripció dels principals components de l'aplicació.

Capítol 4: Disseny tècnic

- Arquitectura de l'aplicatiu: descripció general dels elements que componen l'aplicació i el seu rol dins el sistema.

- Eines: descripció dels llenguatges de programació, *frameworks* i d'altres tecnologies que s'han utilitzat per al desenvolupament i desplegament de l'aplicació.

Capítol 5: Implementació

- Organització del codi: descripció de l'estructura del codi i els paradigmes utilitzats, tant al client com al servidor.
- Explicació en detall d'apartats de codi d'especial interès.

Capítol 6: Proves

- Proves: descripció del tipus de proves implementades, així com les tecnologies utilitzades.

Capítol 7: Revisió de la planificació

- Control de seguiment del progrés del projecte envers la planificació.

Capítol 8: Demostració

- Requisits i procés d'instal·lació: descripció de l'entorn i explicació pas a pas de com executar l'aplicació.
- Demostració de producte: descripció visual de les funcionalitats implementades.

Capítol 9: Conclusions

- Conclusions sobre el projecte, el procés i els resultats obtinguts.

2. Anàlisi

2.1. Anàlisi de mercat

El mercat de la mobilitat és un dels sectors amb més activitat i innovació dels últims temps. La tensió entre l'augment de la densitat de població a les grans ciutats[5] i els alts nivells de pol·lució ha provocat el sorgiment de moltes iniciatives enfocades a donar alternatives de transport.

Una forma de classificar-les seria segons el seu àmbit d'actuació, així tindríem urbanes i interurbanes. Dins el primer àmbit, la ciutat, principalment trobem el lloguer de patinets, bicicletes i motocicletes, així com els vehicles amb conductor (*Uber, Cabify)*. En l'àmbit interurbà, podem trobar des de l'anomenat *car pooling* (compartir places lliures d'un cotxe particular) així com el lloguer d'automòbils per hores. La proposta d'aquest treball s'emmarca dins la categoria de *car pooling*, la qual es troba liderada notablement per l'aplicació ***BlaBlaCar***. Aquesta aplicació és d'origen francès i té presència a multitud de països. L'objectiu és compartir les places lliures de viatgers qui realitzen llargs desplaçaments, normalment per motius de feina o recurrents (per exemple, anar a veure la família quan es viu a una ciutat diferent). D'aquesta manera els usuaris comparteixen els elevats costos de viatjar amb cotxe, els quals inclouen principalment els peatges i el combustible. Tot i que l'aplicació no impedeix l'ús per trajectes diaris, el procés és difícil per a l'usuari, ja que la interfície i el model de dades no està dissenyat per a aquest cas d'ús [6].

Per tant, la conclusió és que actualment no existeix al mercat una aplicació que estigui orientada específicament al *car pooling* de caràcter laboral. Per aquest motiu, l'oportunitat per aquesta aplicació té un gran potencial, però alhora un gran risc, ja que es desconeix el grau d'acceptació per part dels usuaris.

2.2. Enfocament del producte i usuaris potencials

En primer lloc, l'aplicació es dirigeix principalment a usuaris que es desplacen a una ciutat diferent per tal d'acudir a la feina, típicament d'una ciutat als afores cap a la capital. La franja d'edat es podria fixar entre 18 i 45 anys, i es tracta de persones de mentalitat oberta, urbanes i còmodes amb el medi digital.
Com a tota plataforma de tipus *marketplace*, d'una banda trobem els proveïdors, i de l'altra els consumidors. El primer grup serien els **conductors**,

els quals disposen de vehicle propi i es desplacen a la feina a través de vies interurbanes. L'ús del vehicle enfront d'altres mitjans de transport pot ser degut a la comoditat; amb factors com la flexibilitat horària, la realització d'un trajecte directe i, potencialment, més curt, ambientals (temperatura, espai personal, música...), o bé per necessitat: carència d'alternatives de transport públic, necessitat de transportar equips de treball o mercaderies, etc. El fet d'utilitzar l'automòbil per desplaçar-se té una càrrega econòmica associada significativa, des del cost del vehicle (normalment, una quota mensual), passant per l'ineludible cost del combustible i l'assegurança, fins a l'accés a un aparcament, i peatges. El segon grup serien els **usuaris** d'altres mitjans de transport, els quals o bé no disposen de vehicle, o bé no volen assumir els costos associats amb el seu ús (esmenats anteriorment). L'ús de transport públic té també un cost associat gens menyspreable i en constant augment[7], i a més, comporta una sèrie d'inconvenients, tals com la necessitat de desplaçar-se a les estacions, i en molts casos fer transbords, factors ambientals (temperatura, massificació, impossibilitat de seure, etc.) i les avaries, les quals generen una alta frustració als afectats.

2.3. Necessitats dels usuaris

Per tal d'analitzar les necessitats, dividim els usuaris potencials descrits anteriorment en tres grups:

A. **Conductors per decisió pròpia**
B. **Conductors per necessitat**
C. **Usuaris de transport públic**

És important distingir entre aquests grups, ja que en aquest moment no sabem quin tipus d'intercanvi es pot produir. Hi ha dues hipòtesis principals, la primera és que una part dels usuaris de transport públic estarien disposats a assumir un cost sensiblement més alt per tal de millorar els seus desplaçaments diaris, és a dir, gaudir dels beneficis de viatjar en un vehicle privat, sense assumir els costos fixos. Tampoc s'ha de menysprear el factor de la consciència mediambiental. La segona hipòtesi és que hi ha conductors per necessitat que estarien disposats a prescindir de l'ús del seu vehicle si hi hagués una alternativa viable. La mesura en què aquestes dues hipòtesis es produeixin determinaria la magnitud de la disminució de vehicles a les vies d'accés, i per tant, la reducció de la contaminació.

Els **conductors per decisió pròpia** necessiten una manera de reduir el seu impacte mediambiental, ja que és un tema que els preocupa, però alhora no volen renunciar a l'ús del seu vehicle. A més, necessiten una font d'ingressos

per tal de reduir l'alt cost associat amb aquest tipus de desplaçaments. Finalment, també necessiten connectar amb la xarxa de professionals localment i potencialment, comptar amb alternatives de transport en cas que no puguin utilitzar el seu vehicle per qualsevol motiu (avaria, etc.).

Els **conductors per necessitat**, valgui la redundància, també necessiten una font d'ingressos per tal de reduir l'alt cost associat l'ús del seu vehicle, així com una alternativa de transport que els permeti accedir al seu lloc de treball en un temps raonable.

Els **usuaris de transport públic** necessiten una alternativa de transport que millori la seva experiència en relació amb els factors esmenats a l'apartat anterior, sense assumir els alts costos d'utilitzar un automòbil propi.

2.4. Contexts d'ús

A continuació es descriuen diferents situacions on els usuaris utilitzaran les funcionalitats de l'aplicació:

1) Un conductor vol posar a disposició dels usuaris places dins el seu vehicle pels seus desplaçaments diaris. En primer lloc s'haurà de donar d'alta a l'aplicació, introduint tant les seves dades com les del vehicle. A continuació haurà de definir els seus trajectes. Se li presentarà un mapa amb els punts de trobada disponibles i l'usuari podrà triar el seu punt de sortida i d'arribada, així com punts intermedis compatibles amb la recollida d'usuaris. Un cop aquest hagi definit la seva ruta, haurà de definir els seus horaris d'anada i tornada, i si vol mostrar-se disponible automàticament o decidir dia a dia de manera manual. Un cop el registre és complet el conductor és visible a la cerca i està llest per acceptar peticions.

2) Un usuari vol programar un desplaçament. Es dirigirà a l'apartat de *desplaçaments* on se li mostraran conductors compatibles, basant-se en la seva ruta i horari. L'usuari podrà realitzar sol·licituds a 3 conductors diferents. Arribats a aquest punt, l'usuari haurà d'esperar la confirmació d'alguna de les seves sol·licituds. Si passés un temps sense rebre resposta, l'usuari podrà cancel·lar les seves sol·licituds i realitzar-ne de noves. Un cop rep una confirmació, la resta de sol·licituds queden anul·lades i el trajecte queda confirmat.

3) Un conductor vol programar un desplaçament amb un altre conductor. Aquest context és similar a l'anterior, però el conductor ha de realitzar un pas previ: anul·lar la seva disponibilitat per aquell trajecte que vol programar, és a dir, un conductor no podrà oferir places al seu vehicle i alhora realitzar sol·licituds a altres vehicles. Un cop se situa com a no disponible, deixarà d'aparèixer a la cerca per aquell trajecte i podrà realitzar sol·licituds a altres conductors.

4) Un usuari vol conèixer els conductors que té disponibles al seu voltant. L'usuari es dirigirà a l'apartat de *networking* i se li presentarà un conjunt de conductors compatibles. L'usuari veurà la informació principal de cada conductor i podrà dirigir-se al seu perfil per veure'n més detalls. Finalment, per a cada conductor haurà de decidir si vol connectar amb ell o no, tal com succeeix a les aplicacions de *dating* (notablement, Tinder). Els conductors seleccionats rebran una notificació i realitzaran el mateix procés a la inversa. Si ambdós usuaris volen connectar mútuament, es produirà una coincidència i l'aplicació diferenciarà als usuaris a la interfície un cop aquests interactuïn entre si.

2.5. Fitxes d'usuaris i escenaris

Conductor per decisió pròpia concernit per la contaminació

Sergi
33 anys
Enginyer industrial
Casat
Sabadell

Perfil
Sergi és casat amb l'Helena i tots dos viuen al barri de la Creu Alta de Sabadell. És enginyer industrial per la UAB i treballa a Indra, concretament a la seva oficina dins el districte 22@ de Barcelona.

Va ser usuari de tren (FGC) en els seus anys d'estudiant i també durant les seves primeres experiències professionals. Un cop assoleix la seva posició actual, decideix adquirir un nou automòbil i utilitzar-lo per anar a la feina. Pensa que amb el transport públic perd molt de temps i vol gaudir de la comoditat de viatjar en el seu vehicle.

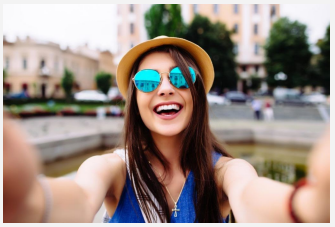
Tot i això, és conscient que aquest canvi li suposa uns costos afegits, com el combustible i la plaça d'aparcament que ha de llogar, però creu que amb els seus ingressos actuals pot absorbir-los.

A causa dels últims desenvolupaments en relació amb la contaminació, tals com la declaració de les ZBE (zones de baixes emissions) i els continuats episodis climàtics extrems, és conscient que l'ús del vehicle té un gran impacte i se sent corresponsable, però de moment considera que no hi ha una alternativa viable, i en cap cas tornaria a utilitzar el transport públic mentre mantingui el mateix nivell d'ingressos.

Escenari d'ús
La setmana passada va decidir descarregar-se l'aplicació Ronda i donar-se d'alta. Pensa que pot disminuir el seu impacte mediambiental i alhora tenir uns ingressos addicionals que cobriran gran part dels costos associats al desplaçament amb cotxe. Li preocupa la disrupció en relació amb la recollida dels passatgers i el perfil d'aquests.
El diumenge l'aplicació el notifica que té diverses sol·licituds de passatgers. Dedica uns minuts a mirar els perfils i accepta dues, una d'anada i una de tornada.
El dilluns obre l'aplicació i comença el trajecte cap a la feina. Un cop arriba al punt de trobada, el passatger l'està esperant, s'atura i aquest puja al

vehicle. Durant el trajecte conversa amb el passatger i a l'arribada, els dos es baixen i es dirigeixen als seus llocs de treball. Ja dins l'oficina, obre l'aplicació i marca el trajecte com a completat. Expressa que l'experiència li ha resultat agradable i el passatger ha estat puntual. Quan acaba, el passatger ja ha confirmat el trajecte i rep els fons a la seva compta.

Conductora per necessitat preocupada pel cost de transport



Júlia
29 anys
Comunicació
Soltera
Sant Quirze

Perfil
Júlia és soltera i viu al centre de Sant Quirze. És consultora de comunicació i experta en xarxes socials, i treballa a una consultoria de màrqueting *online* situada a Diagonal Mar.

Sempre ha sigut partidària d'utilitzar transport públic i està a favor de les noves formes de mobilitat, però actualment li resulta inviable arribar al seu lloc de treball amb transport públic en un temps raonable, ja que no hi ha una opció directa. Hauria de desplaçar-se en cotxe per arribar a una estació de la RENFE, i després agafar un autobús.

Està conscienciada, però no està disposada a perdre més de dues hores diàries en desplaçar-se a la feina. Per aquest motiu ha arribat a la conclusió que ha d'utilitzar el seu automòbil.

Li preocupa especialment l'alt cost que suposa desplaçar-se amb el seu vehicle; tot i que no li cal llogar una plaça d'aparcament, el preu del combustible és cada cop més alt, i la quota de l'assegurança ha augmentat com a conseqüència en l'augment del quilometratge. A més, ha hagut de realitzar 2 reparacions l'últim any, per les quals ha dedicat una part dels estalvis que tenia reservats per anar-se'n de viatge a l'estiu.

Escenari d'ús
Són les 19.00 hores d'un dijous i vol reservar un trajecte pel dia de demà amb un conductor, ja que després ha quedat amb unes amigues i tornarà a casa amb una d'elles, per la qual cosa pot deixar el cotxe.
Obre l'aplicació i es dirigeix a la secció de desplaçaments. Dedica uns minuts a examinar els conductors disponibles. Finalment troba un parell

d’opcions que li encaixen i realitzar la sol·licitud.
Han passat 30 minuts i rep la notificació que una de les seves sol·licituds ha estat aprovada.
L'endemà es dirigeix al punt de trobada i espera l’arriba del vehicle, del qual sap la matrícula, marca, model i color. Pocs minuts després, apareix el conductor i després d’una breu salutació pugen al cotxe.
40 minuts després arriben al punt desitjat i s’acomiada del conductor, ja que aquest encara continua amb el seu trajecte. Ja dins l’oficina, obre l’aplicació i marca el trajecte com a completat. Expressa que l’experiència li ha resultat agradable i que el conductor ha estat puntual.

Usuària de transport públic insatisfeta

Ariadna
34 anys
Recursos Humans
Casada
Sabadell

Perfil

Ariadna és casa amb el Toni i tots dos viuen al barri de Gràcia de Sabadell. És psicòloga per la UB i treballa com a consultora de recursos humans a una agència de treball temporal amb oficina al Passeig Sant Joan de Barcelona.

Sempre ha utilitzat el transport públic, especialment la línia Barcelona Sants - Manresa de RENFE. Actualment continua utilitzant aquesta línia per anar a la feina. Tant la parada d’anada com de tornada li queden a prop de casa i de la feina, respectivament, per la qual cosa està relativament satisfeta en allò que respecta a la conveniència.

Tot i això, hi ha dies en què es pregunta si hi ha alguna alternativa. Les avaries i retardaments són freqüents, i si no està atenta durant el trajecte, normalment no aconsegueix asseure’s. Ara per ara no li suposa un gran problema, però té dubtes sobre aquesta dinàmica diària a llarg termini. Més d’una vegada ha jugat amb la idea de comprar-se un cotxe, però l’alt cost que suposaria la fa dubtar.

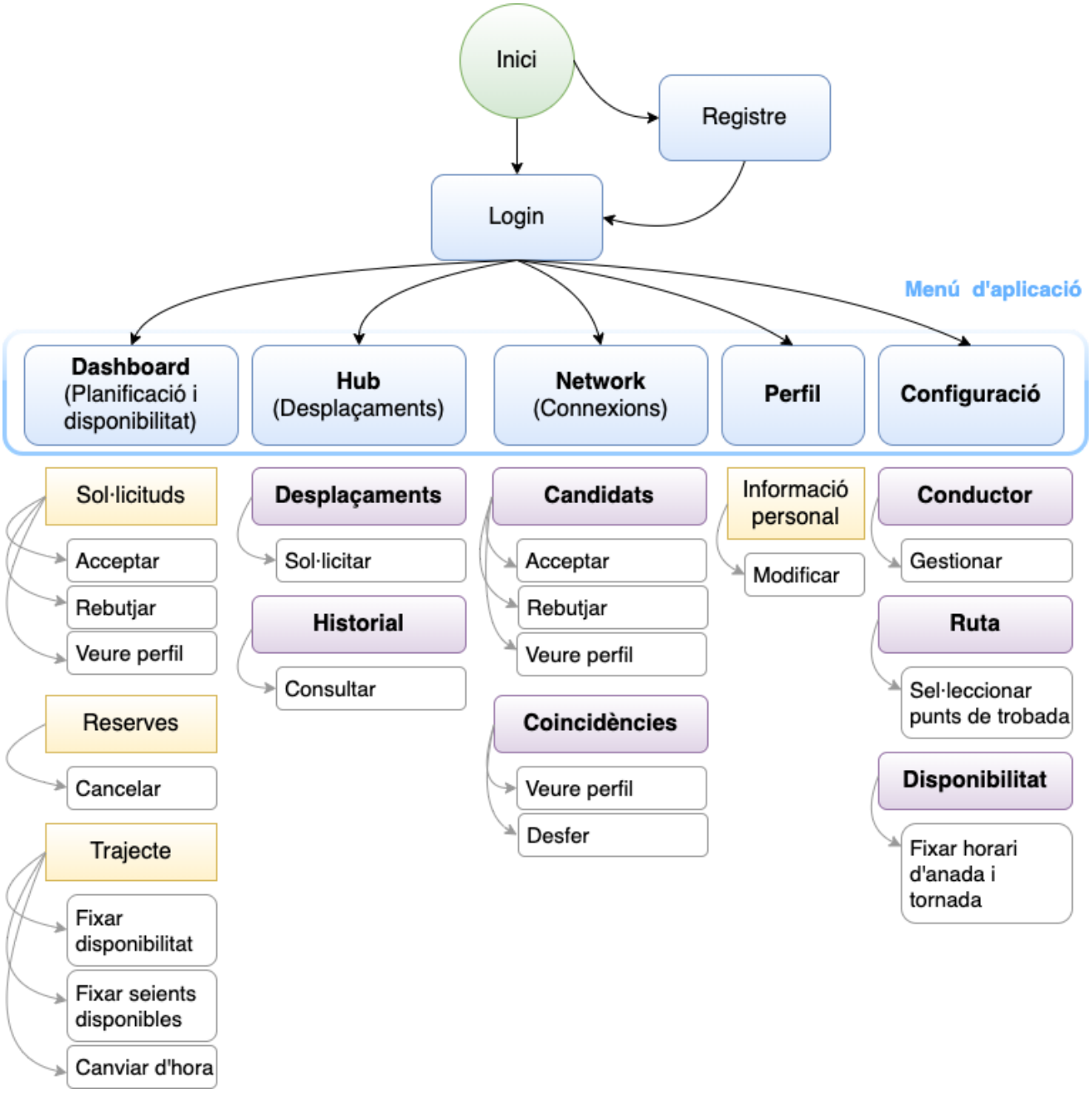
Escenari d’ús
És dimarts nit, després de sopar es troba asseguda al sofà descansant. Avui el tren s’ha retardat per una avaria i ha arribat mitja hora tard a la feina, temps que ha hagut de recuperar. Demà té una classe després de la feina

així que vol assegurar-se que podrà sortir a la seva hora, per la qual cosa decideix obrir l'aplicació i veure quines opcions hi ha per demà. Dedica un temps a mirar els perfils dels conductors disponibles i envia 3 sol·licituds.
En pocs minuts rep la notificació que una de les sol·licituds ha estat acceptada.
Al matí es dirigeix al punt de recollida amb uns minuts d'antelació i espera l'arribada del vehicle. Un cop identifica el cotxe que espera, s'identifiquen mútuament i es puja. Durant el trajecte conversa amb el conductor, el qual treballa també dins recursos humans, i comparteixen experiències professionals. Al punt d'arribada es baixa del cotxe i un cop es troba a l'oficina obre l'aplicació i confirma el trajecte. Decideix guardar el conductor com a connexió dins l'aplicació per tal de tornar a repetir amb ell en futurs trajectes.

3. Disseny i experiència d'usuari

3.1. Arbre de navegació

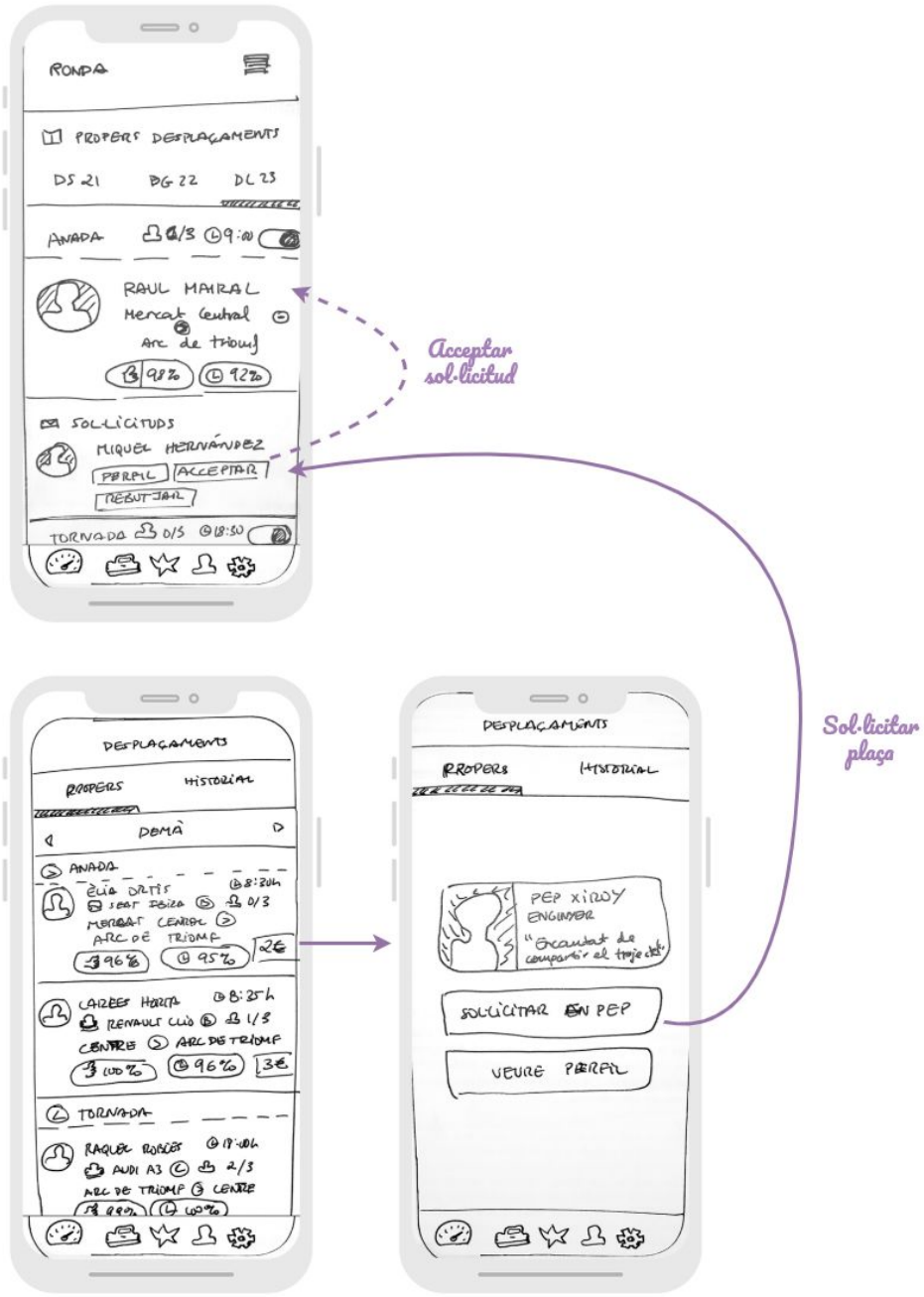
Un cop analitzades les necessitats dels usuaris, els contexts d'ús i de disposar dels objectius funcionals definits a l'apartat 1.2 Objectius del Treball, a continuació es defineix l'arbre de navegació de l'aplicació. Tot i que la implementació es pot veure en el següent apartat, és adient mencionar que la navegació principal es realitzarà amb un menú inferior que donarà accés a les cinc seccions principals (blau), i dins una secció s'utilitzaran pestanyes a la part superior per accedir a les subseccions (lila).

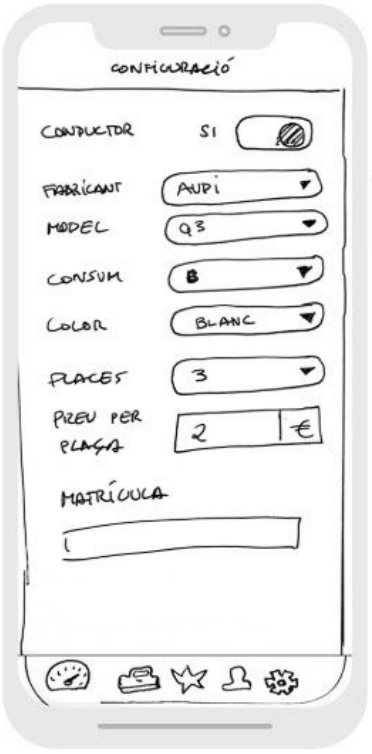


3.2. Prototip

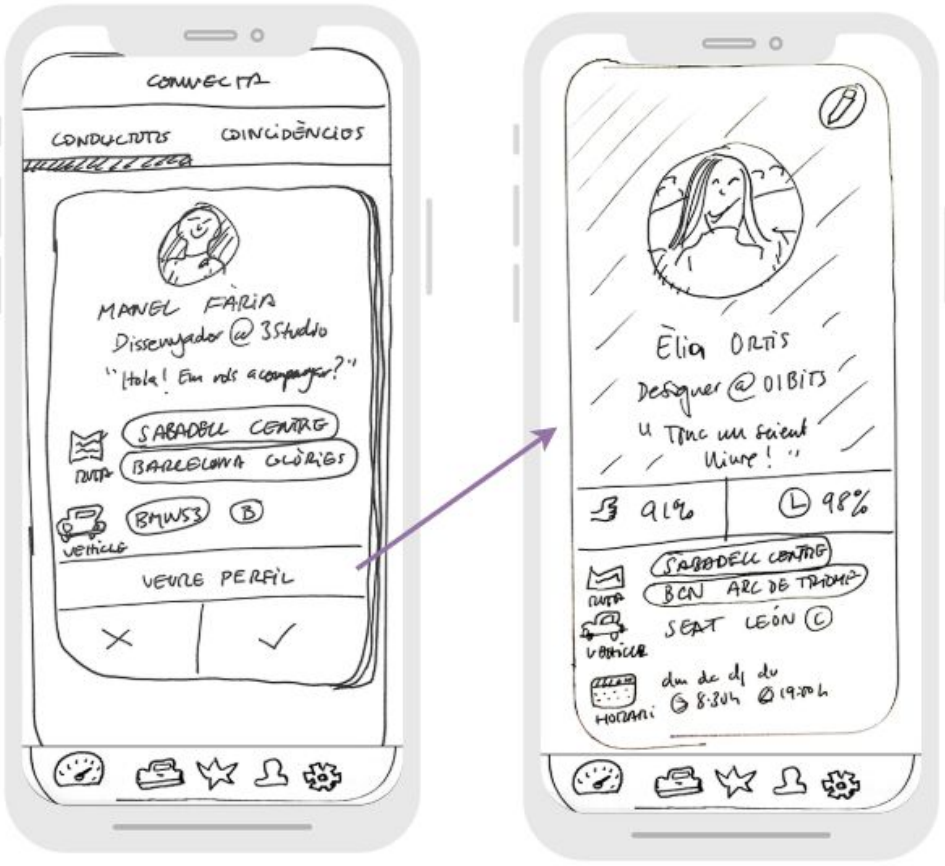
Un cop realitzat el disseny conceptual, amb la definició d'escenaris d'ús i els fluxos d'interacció, es dedica la següent secció al prototipatge. Aquest procés es dividirà en 2 fases, d'una banda es realitzarà un prototip a mà (*sketching*) per tal de concretar el disseny i iterar sobre diferents alternatives de manera ràpida. A continuació s'inclourà un prototip d'alta fidelitat, realitzat amb l'eina *Justinmind*.

3.2.1. Sketching

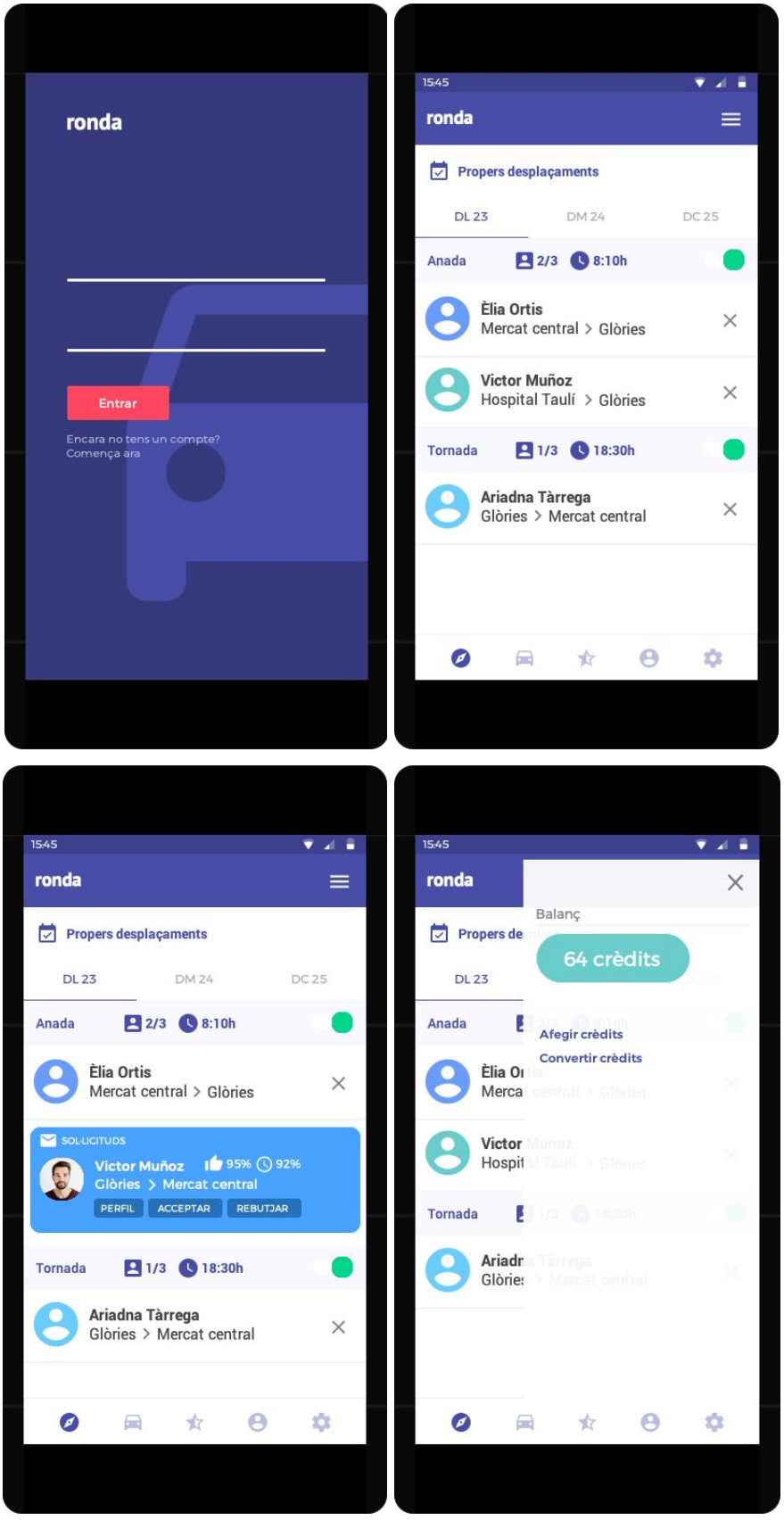


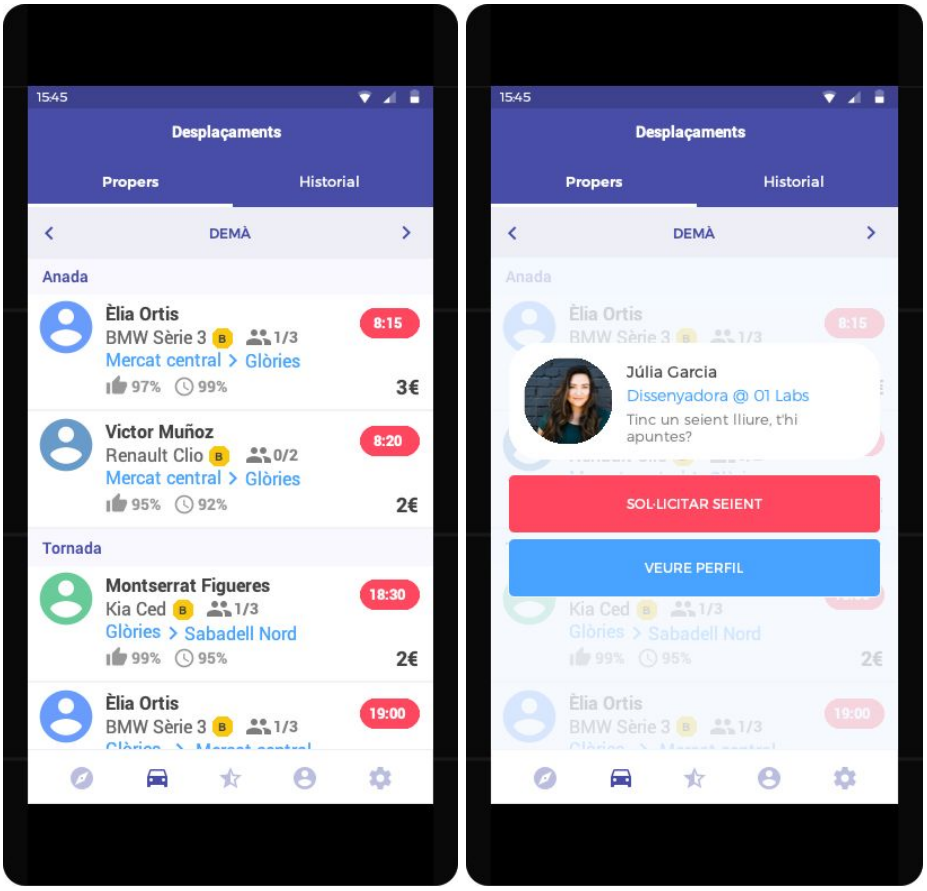


Sel·lecció punt de trobada

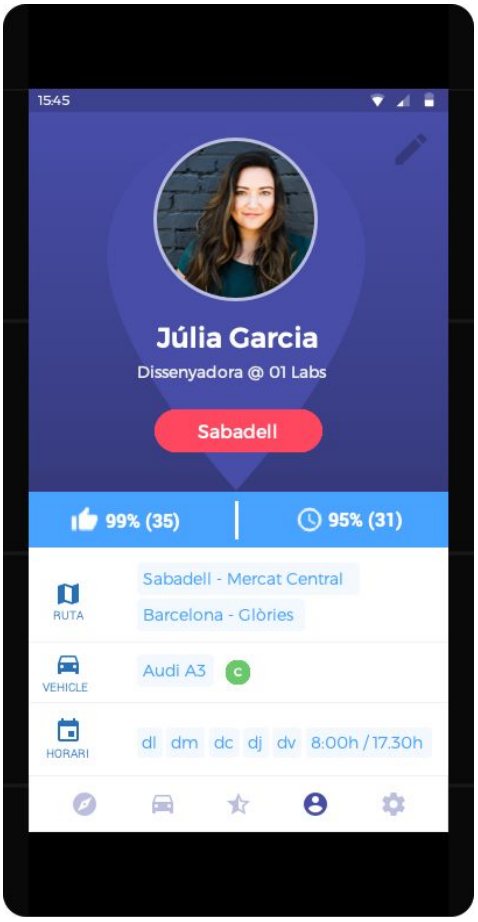

CONNECTA
CONDUCTORS
COINCIDÈNCIES
MANEL FARIA
Dissenyador @ 3Studio
"Hola! Em vols acompanyar?"
SABADELL CENTRE
BARCELONA GLÒRIES
BMW33
B
VEURE PERFIL
Èlia ORTIS
Desginer @ OIBITS
U Tinc un seient lliure!"
91%
98%
SABADELL CENTRE
BCN ARC DE TRIOMF
SEAT LEÓN C

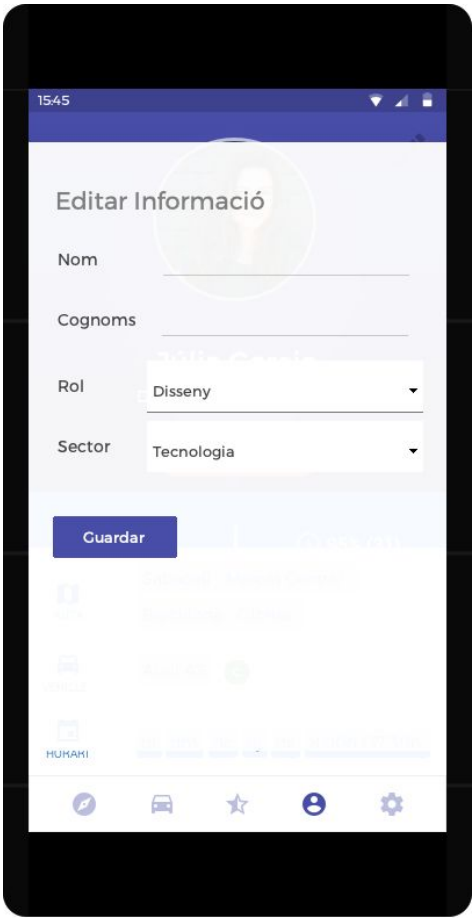
3.2.2. Prototip horitzontal d'alta fidelitat

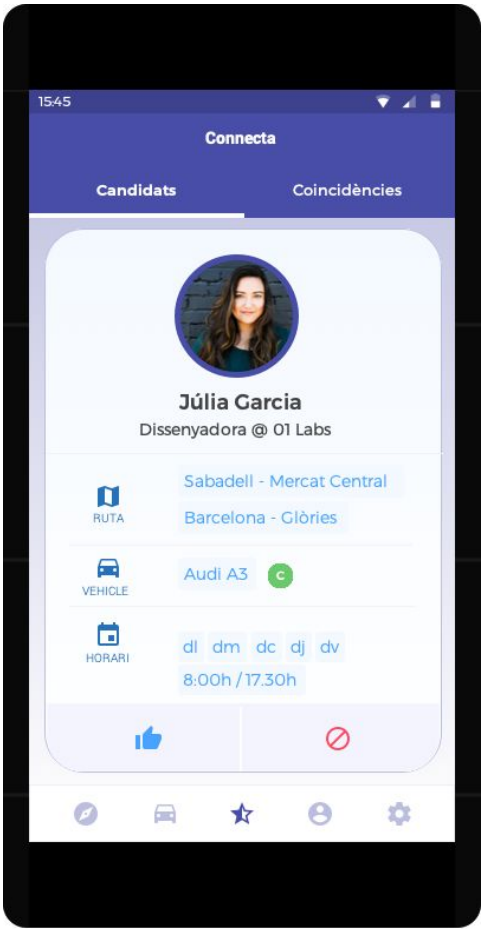


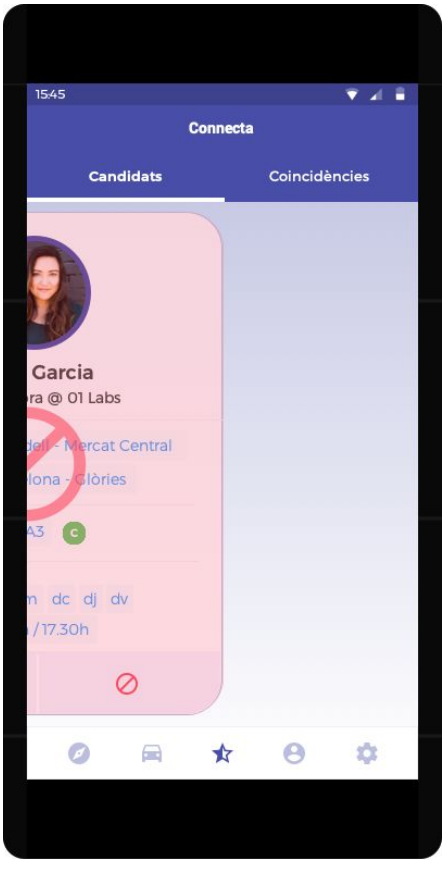

15:45
Desplaçaments
Propers
Historial
DEMÀ
Anada
Èlia Ortis
BMW Sèrie 3
1/3
Mercat central > Glòries
97%
99%
3€
8:15
Victor Muñoz
Renault Clio
0/2
Mercat central > Glòries
95%
92%
2€
8:20
Tornada
Montserrat Figueres
Kia Ced
1/3
Glòries > Sabadell Nord
99%
95%
2€
18:30
Èlia Ortis
BMW Sèrie 3
1/3
19:00

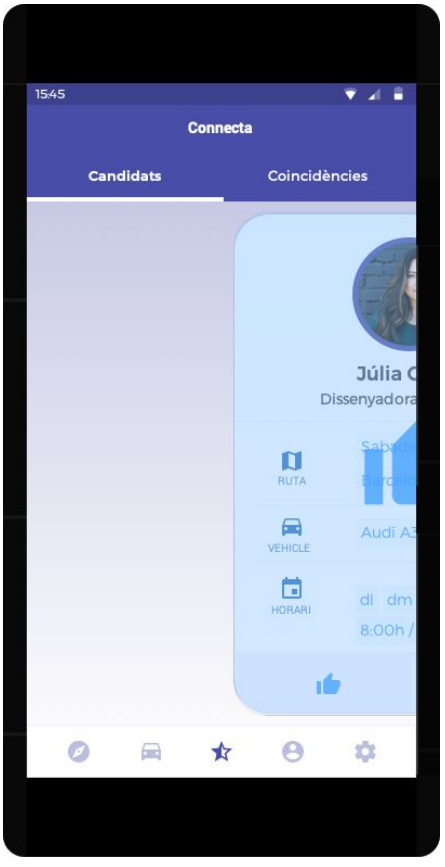
15:45
Desplaçaments
Propers
Historial
DEMÀ
Anada
Júlia Garcia
Dissenyadora @ 01 Labs
Tinc un seient lliure, t'hi apuntes?
SOL·LICITAR SEIENT
VEURE PERFIL

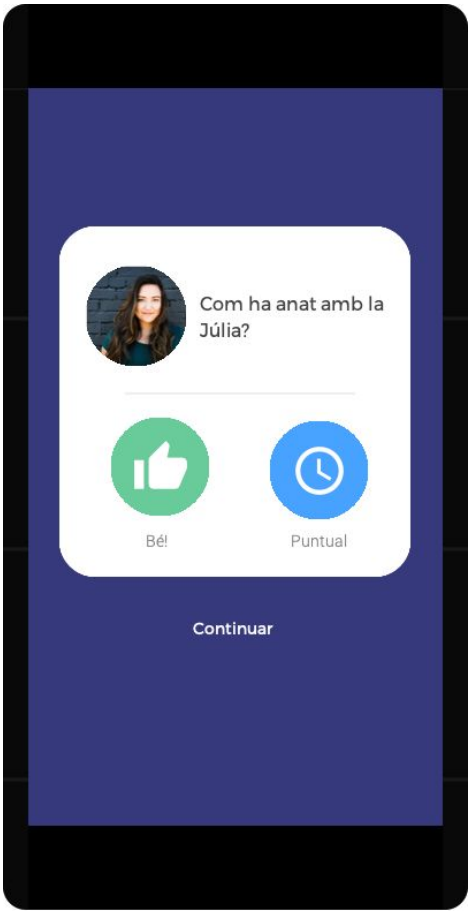

15:45
Júlia Garcia
Dissenyadora @ 01 Labs
Sabadell
99% (35)
95% (31)
RUTA
Sabadell - Mercat Central
Barcelona - Glòries
VEHICLE
Audi A3
HORARI
dl dm dc dj dv 8:00h / 17.30h

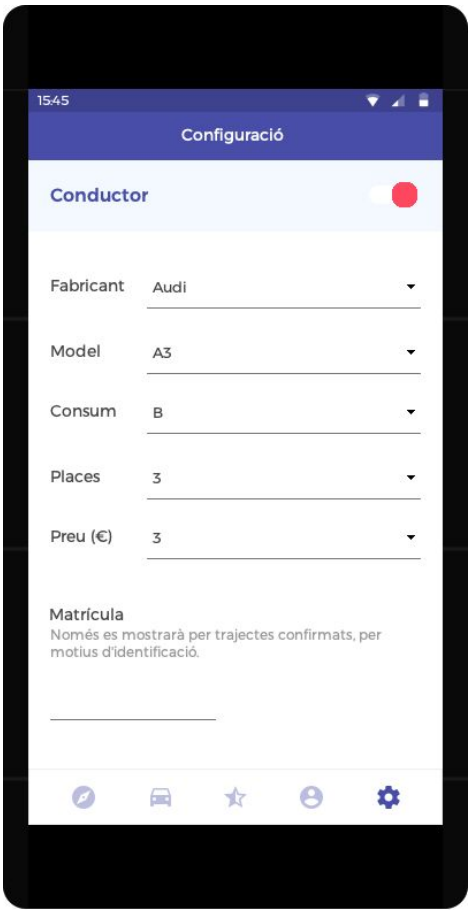

15:45
Editar Informació
Nom
Cognoms
Rol
Disseny
Sector
Tecnologia
Guardar

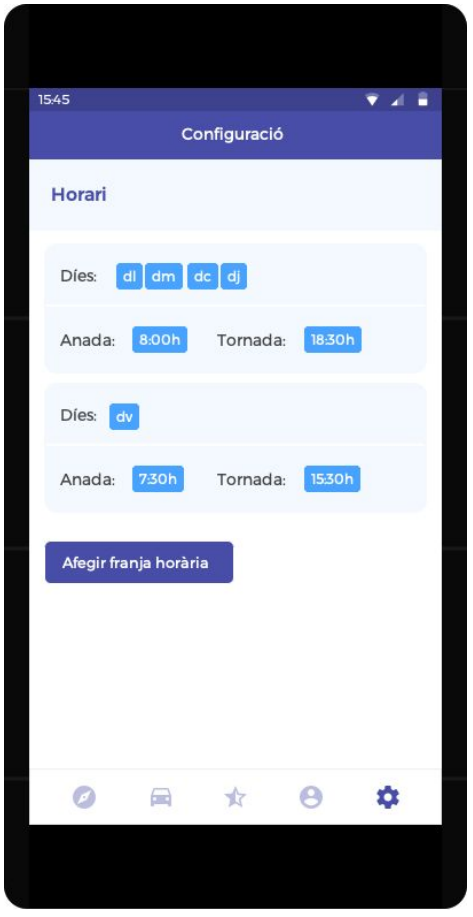

15:45
Connecta
Candidats
Coincidències
Júlia Garcia
Dissenyadora @ 01 Labs
RUTA
Sabadell - Mercat Central
Barcelona - Glòries
VEHICLE
Audi A3
HORARI
dl dm dc dj dv
8:00h / 17.30h

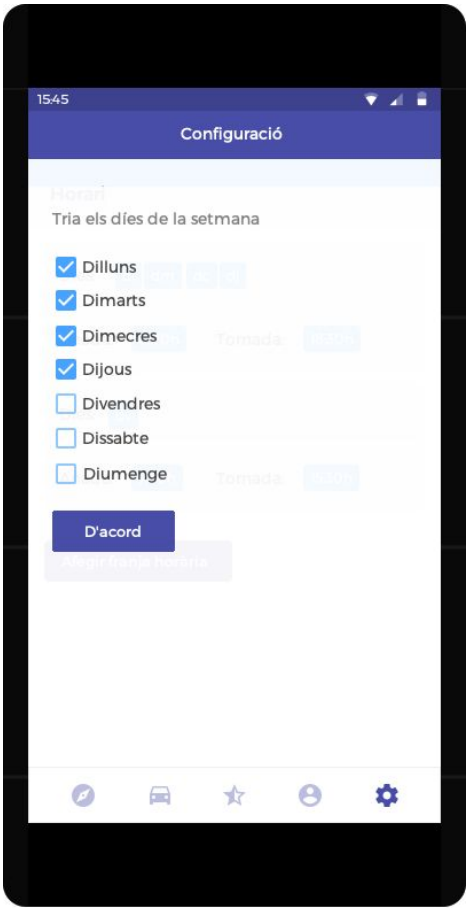

15:45
Connecta
Candidats
Coincidències

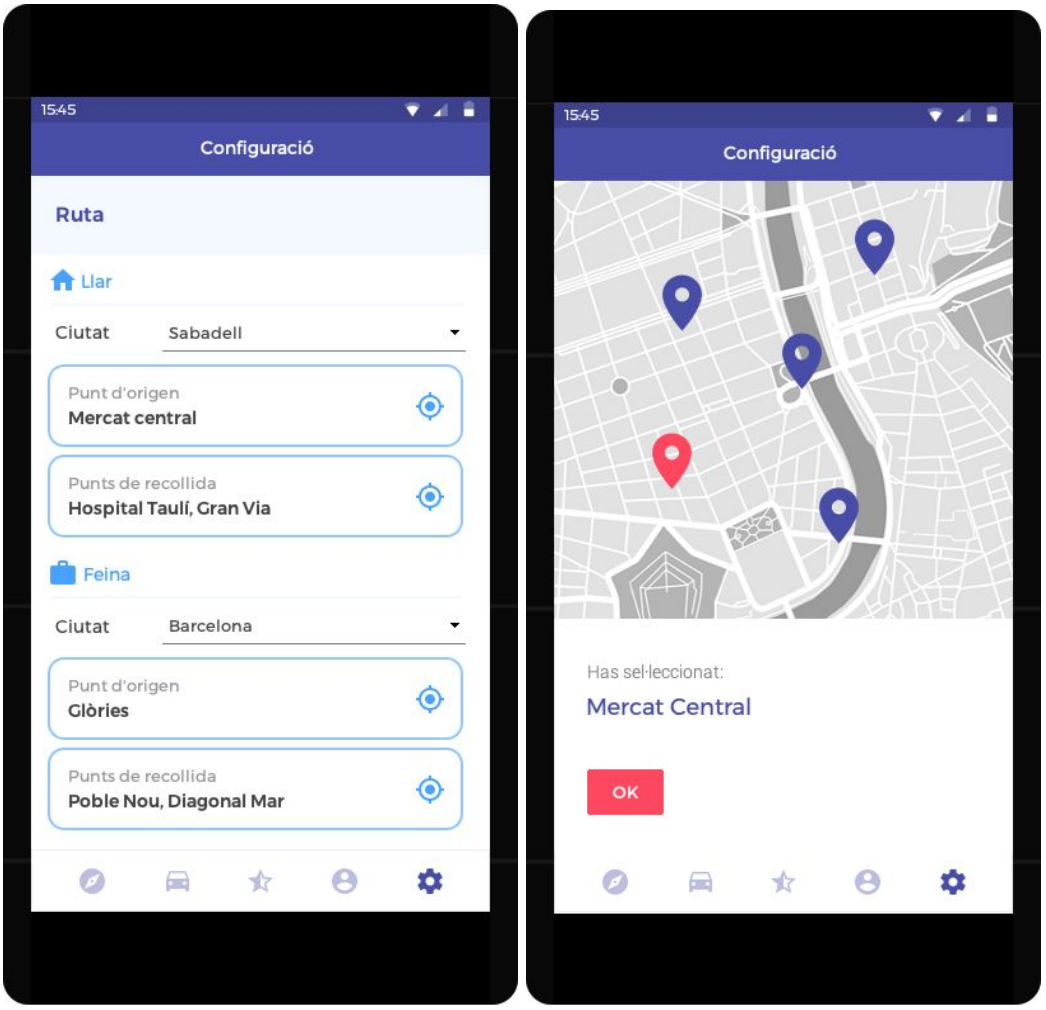

15:45
Connecta
Candidats
Coincidències


Com ha anat amb la Júlia?
Bé!
Puntual
Continuar


15:45
Configuració
Conductor
Fabricant
Audi
Model
A3
Consum
B
Places
3
Preu (€)
3
Matrícula
Només es mostrarà per trajectes confirmats, per motius d'identificació.


15:45
Configuració
Horari
Díes:
dl
dm
dc
dj
Anada:
8:00h
Tornada:
18:30h
Díes:
dv
Anada:
7:30h
Tornada:
15:30h
Afegir franja horària

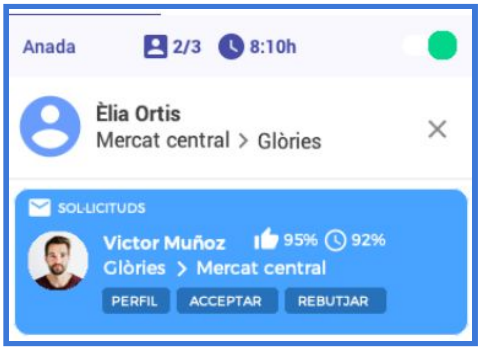

15:45
Configuració
Tria els díes de la setmana
Dilluns
Dimarts
Dimecres
Dijous
Divendres
Dissabte
Diumenge
D'acord


15:45
Configuració
Ruta
Llar
Ciutat
Sabadell
Punt d'origen
Mercat central
Punts de recollida
Hospital Taulí, Gran Via
Feina
Ciutat
Barcelona
Punt d'origen
Glòries
Punts de recollida
Poble Nou, Diagonal Mar
15:45
Configuració
Has sel·leccionat:
Mercat Central
OK

3.2.3. Comentari sobre aspectes singulars de la proposta de disseny

Agenda

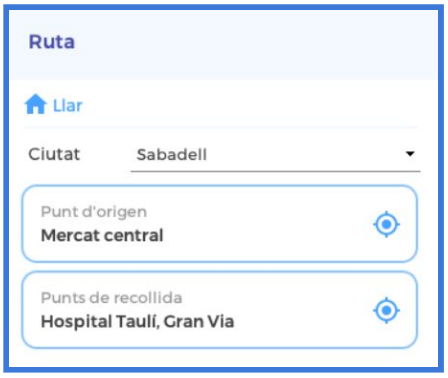
Una de les principals peculiaritats en l'àmbit de la interfície es dóna a la secció inicial, l'agenda (*propers desplaçaments*). En primer lloc, s'ha decidit limitar la interfície a tres dies. Això és degut a la volatilitat existent de la disponibilitat d'aquests trajectes. Problemes mecànics, reunions extraordinàries a la feina o visites mèdiques són només algunes de les múltiples causes que poden suposar la modificació o cancel·lació d'un trajecte, tant d'una part com de l'altra. Per tant, dit d'una altra manera, no resultaria útil realitzar una reserva 10 dies en endavant, quan la probabilitat que aquesta canviï o no es produeixi és significativa.



Un altre aspecte important és que la interfície determina clarament un trajecte d'anada i un de tornada. Per tant l'aplicació no es podrà utilitzar per realitzar múltiples trajectes en un mateix dia, evitant així usuaris temptats a utilitzar l'aplicació per realitzar serveis de transport, i per tant, evitant cap mena de conflicte amb el sector del taxi.

Tal com s'esmena al punt 1.3 Enfocament i mètode seguit un dels obstacles que, fins ara, han tingut aquest tipus d'aplicacions és la necessitat de coordinar un punt de trobada. Normalment els conductors publiquen trajectes i els usuaris troben de manera recurrent aquell(s) que li resulten apropiats. Aquest procés requereix un esforç constant a l'hora de trobar trajectes i d'establir punts de trobada entre conductors i passatgers.

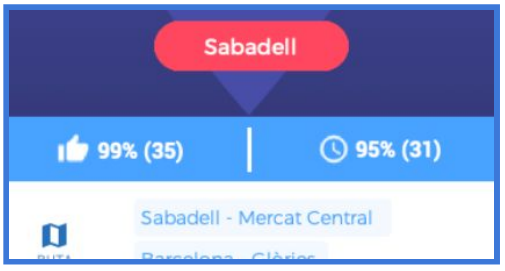
Punts de trobada



Com també s'esmena en el punt 1.3, es considera com una de les claus del projecte el fet de reduir al mínim l'esforç necessari perquè els interessats interactuïn. Aquesta idea ha portat a capgirar el plantejament tradicional basat en el model *p2p* (*peer to peer*) i GPS, i utilitzar punts de trobada fixos, anàlegs a les estacions de transport públic. És a dir, el conductor *dóna servei* a una *línia* de transport concreta, la qual es defineix de la següent manera: 1) punt de sortida, 2) punt(s) intermedis i 3) punt d'arribada.

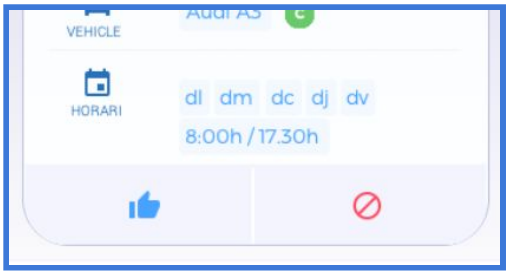
Els intermedis són aquells punts situats entre l'origen i l'arribada per on passa la ruta. D'aquesta manera s'aconsegueix, d'una banda, que els usuaris no hagin d'interactuar per tal de trobar-se (els punts de trobada són coneguts amb antelació) i, d'altra banda, s'expandeix l'abast del conductor gràcies als punts intermedis.

Estadístiques



En diferents seccions de l'aplicació apareixen dues estadístiques associades a cada usuari: *"m'agrades"* (mà amb el polze aixecat) i puntualitat (rellotge). Un cop es completa el trajecte, tant el conductor com el passatger tenen l'oportunitat de donar feedback l'un a l'altre a través d'aquestes dues variables. D'aquesta manera s'incentiva als usuaris, ja que les estadístiques afectaran les dues parts a l'hora de realitzar sol·licituds i acceptar-les.

Connexions



S'ha inclòs una secció per tal que els usuaris puguin contactar entre ells. S'ha utilitzat el paradigma del desplaçament de targetes que va popularitzar l'aplicació *Tinder*. L'objectiu d'aquesta secció és introduir un element d'interacció per aportar un valor afegit i anar més enllà de ser únicament una "eina", per assimilar-se a una xarxa social on treballadors que es desplacen connecten entre ells (un *LinkedIn* local*).*

3.3. Avaluació

3.3.1. Preguntes sobre l'usuari

- Informació bàsica: edat, professió
- Localització de la llar i el lloc de treball
- Horari laboral
- Mètode de transport i vehicle, en cas de tenir-ne
- Disponibilitat i/o motivació a compartir el vehicle

3.3.2. Tasques que hauria de realitzar

1. Crear un compte
2. Introduir informació sobre el vehicle (si en té), horari laboral i trajectes
3. Sol·licitar un trajecte / Acceptar una sol·licitud
4. Realitzar una connexió amb un altre usuari
5. Visitar el perfil d'un altre usuari

3.3.3. Preguntes referents a les tasques

1. Has trobat alguna dificultat a l'hora del registre?
2. La quantitat d'informació demandada t'ha semblat adequada, insuficient o excessiva?
3. Has pogut reflectir amb precisió la informació relativa al vehicle, horaris i trajectes?
4. A l'hora de sol·licitar un trajecte / d'acceptar una sol·licitud; has pogut obtenir tota la informació necessària per fer el pas? Has trobat alguna dificultat en la realització de la tasca?
5. A l'hora de revisar els candidats i perfils d'usuari, consideres que la presentació t'ha facilitat la consulta o has trobat algun obstacle?

4. Disseny tècnic

4.1. Definició de casos d'ús

CU-01	Registre
Descripció	Crear un compte d'usuari per poder accedir a l'aplicació.
Prioritat	Alta
Precondicions	L'usuari no ha de disposar d'un compte amb la metodologia triada (correu electrònic, telèfon, Google…).
Flux	Un cop s'obre l'aplicació, si l'usuari no ha entrat prèviament, se li mostraran les metodologies per crear un compte. L'usuari escollirà una, introduirà les dades que se li demanin i confirmarà l'operació.
Postcondicions	• Si no existeix un compte previ amb la metodologia utilitzada, es crea el compte i l'usuari accedeix a l'aplicació. • Si existeix un compte previ, es notifica l'usuari i se li dóna l'opció d'iniciar un procés de recuperació de contrasenya.

CU-02	Accedir
Descripció	Accedir a l'aplicació a través d'un procés d'autenticació.
Prioritat	Alta
Precondicions	L'usuari ha de disposar d'un compte.
Flux	Un cop s'obre l'aplicació, si l'usuari no ha entrat prèviament, se li mostraran les metodologies per tal d'autenticar-se. L'usuari triarà la seva, omplirà l'identificador i la contrasenya.
Postcondicions	• Si l'identificador i contrasenya són correctes es redirigirà l'usuari a la pàgina d'inici. • Si les dades no són correctes, es notificarà l'usuari.

CU-03	Introduïr dades de perfil
Descripció	L'usuari introdueix informació sobre el seu perfil, nom i cognom, perfil i sector professional, i una breu biografia o missatge d'estat.
Prioritat	Alta
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament)
Flux	I. Procés d'*embarcament* (primer accés a l'aplicació). II. Secció *perfil*.
Postcondicions	• Es guarden les dades introduïdes o modificades. • Si l'usuari es troba en el procés d'embarcament, segueix al següent pas.

CU-04	Establir disponibilitat
Descripció	L'usuari defineix la seva disponibilitat, la qual consisteix d'un llistat de dies, així com un horari d'anada i tornada.
Prioritat	Alta (conductors) / Mitja
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament)
Flux	III. Procés d'*embarcament* (primer accés a l'aplicació). IV. Secció *configuració*, opció: *disponibilitat*.
Postcondicions	• Es guarden les dades introduïdes o modificades. • Si l'usuari es troba en el procés d'embarcament, segueix al següent pas.

CU-05	Establir ruta
Descripció	L'usuari defineix la seva ruta, la qual consisteix de dos punts de trobada.
Prioritat	Alta (conductors) / Mitja
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament)
Flux	V. Procés d'*embarcament* (primer accés a l'aplicació). VI. Secció *configuració*, opció: *ruta*.
Postcondicions	• Es guarden les dades introduïdes o modificades.

	• Si l'usuari es troba en el procés d'embarcament, segueix al següent pas.

CU-06	Establir dades de conductor
Descripció	L'usuari es defineix com a conductor, i afegeix les dades del vehicle.
Prioritat	Alta (només conductors)
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament)
Flux	VII. Procés d'*embarcament* (primer accés a l'aplicació). VIII. Secció *configuració*, opció: *conductor*.
Postcondicions	• Es guarden les dades introduïdes o modificades. • Si l'usuari es troba en el procés d'embarcament, segueix al següent pas.

CU-07	Sol·licitar trajecte
Descripció	L'usuari escull un conductor per un trajecte dins una ruta i horari determinats.
Prioritat	Alta
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament) CU-0[3-6] (haver completat el procés d'embarcament)
Flux	L'usuari accedeix a la secció de *desplaçaments*, dins un dia determinat, i escull un conductor completant l'acció d'enviar la sol·licitud.
Postcondicions	• Es notifica a l'usuari que la sol·licitud s'ha realitzat correctament. • La sol·licitud enviada es mostra en la secció *agenda*, sota el dia i l'horari corresponents.

CU-08	Acceptar sol·licitud
Descripció	El conductor accepta una sol·licitud rebuda d'un usuari que vol realitzar el trajecte amb ell.
Prioritat	Alta
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament) CU-0[3-6] (haver completat el procés d'embarcament) L'usuari és conductor. CU-07 (haver rebut una sol·licitud)
Flux	L'usuari accedeix a la secció d'*agenda*, dins un dia determinat, i expandeix la secció de sol·licituds, on realitza l'acció d'acceptar.
Postcondicions	• Es notifica a l'usuari que la sol·licitud s'ha acceptat correctament. • La sol·licitud acceptada es mostra en la secció *agenda*, sota el dia i l'horari corresponents. • L'usuari que realitza la sol·licitud rep una notificació de l'acceptació.

CU-09	Rebutjar sol·licitud
Descripció	El conductor rebutja una sol·licitud rebuda d'un usuari que vol realitzar el trajecte amb ell.
Prioritat	Alta
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament) CU-0[3-6] (haver completat el procés d'embarcament) L'usuari és conductor. CU-07 (haver rebut una sol·licitud)
Flux	L'usuari accedeix a la secció d'*agenda*, dins un dia determinat, i expandeix la secció de sol·licituds, on realitza l'acció de rebutjar.
Postcondicions	• Es notifica a l'usuari que la sol·licitud s'ha rebutjat correctament. • La sol·licitud rebutjada desapareix de la secció *agenda*, sota el dia i l'horari corresponents. • L'usuari que realitza la sol·licitud rep una notificació del rebuig.

CU-10	Modificar disponibilitat de trajecte
Descripció	El conductor pot modificar la disponibilitat d'un dia i l'horari concrets.
Prioritat	Mitja
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament) CU-0[3-6] (haver completat el procés d'embarcament) L'usuari és conductor.
Flux	L'usuari accedeix a la secció d'*agenda*, dins un dia determinat, i canvia la seva disponibilitat amb l'interruptor.
Postcondicions	• Es reflecteix el canvi en la interfície. • Si hi havia sol·licituds i/o reserves s'obre un diàleg on l'usuari ha de confirmar l'acció. Un cop confirmada, es cancel·len totes les sol·licituds i reserves existents, i es notifica els usuaris que les van realitzar.

CU-11	Modificar hora de trajecte
Descripció	El conductor pot modificar l'hora d'un trajecte.
Prioritat	Mitja
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament) CU-0[3-6] (haver completat el procés d'embarcament) L'usuari és conductor.
Flux	L'usuari accedeix a la secció d'*agenda*, dins un dia determinat, fa clic a l'hora actual i selecciona l'hora desitjada al rellotge.
Postcondicions	• Es reflecteix el canvi en la interfície. • Si hi havia sol·licituds i/o reserves s'obre un diàleg on l'usuari ha de confirmar l'acció. Un cop confirmada, es cancel·len totes les sol·licituds i reserves existents, i es notifica els usuaris que les van realitzar.

CU-12	Modificar places disponibles
Descripció	El conductor pot modificar les places disponibles d'un trajecte.
Prioritat	Mitja
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament) CU-0[3-6] (haver completat el procés d'embarcament) L'usuari és conductor.
Flux	L'usuari accedeix a la secció d'*agenda*, dins un dia determinat, fa clic al nombre de places actual i incrementa o redueix al nombre desitjat.
Postcondicions	• Es reflecteix el canvi en la interfície. • Si hi havia reserves confirmades, el nou nombre de places no serà inferior que el nombre de reserves.

CU-13	Cancel·lar reserva
Descripció	Tant el conductor com el passatger poden cancel·lar una reserva.
Prioritat	Alta
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament) CU-0[3-6] (haver completat el procés d'embarcament) CU-08 (haver confirmat una reserva)
Flux	L'usuari accedeix a la secció d'*agenda*, dins un dia determinat, i cancel·la la reserva. Es mostra un diàleg on es demana confirmar l'operació.
Postcondicions	• La reserva a cancel·lar desapareix de la secció *agenda*, sota el dia i l'horari corresponents. • L'altre usuari rep una notificació de la cancel·lació.

CU-14	Connectar amb usuari
Descripció	Un usuari decideix connectar amb un altre usuari dins la plataforma.
Prioritat	Baixa
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament) CU-0[3-6] (haver completat el procés d'embarcament)
Flux	L'usuari accedeix a la secció de *connexions*, i desplaça la *carta* de l'usuari corresponent cap a la dreta, o bé fa clic a la icona de confirmació.
Postcondicions	• L'usuari amb qui vol connectar rep una notificació informant-lo sobre el fet. • Quan l'usuari amb qui vol connectar es dirigeixi a la secció *connexions*, l'usuari que realitza l'acció apareixerà primer, si és l'únic interessat en connectar en aquell moment, o en cas contrari, en l'ordre que correspongui segons els usuaris interessats a connectar i la data en què es va realitzar l'acció per part de cadascú.

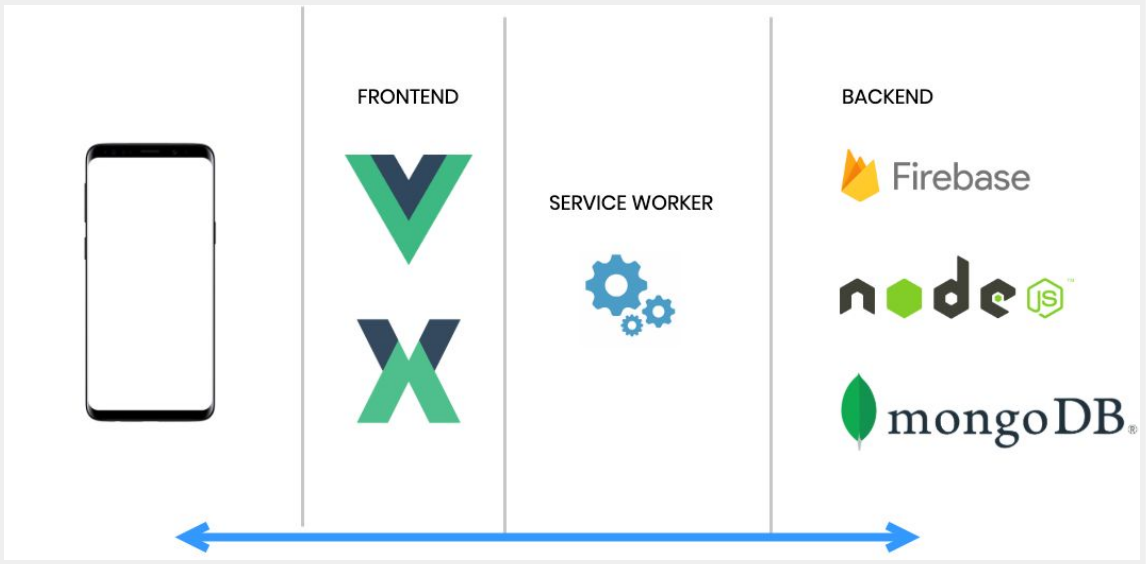
CU-15	Rebutjar connectar amb usuari
Descripció	Un usuari decideix no connectar amb un altre usuari dins la plataforma.
Prioritat	Baixa
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament) CU-0[3-6] (haver completat el procés d'embarcament)
Flux	L'usuari accedeix a la secció de *connexions*, i desplaça la *carta* de l'usuari corresponent cap a l'esquerra, o bé fa clic a la icona de rebutjar.
Postcondicions	• L'usuari el qual s'ha rebutjat no apareix de nou al llistat.

CU-16	Afegir crèdits
Descripció	Un usuari pot afegir crèdits al seu balanç per tal d'intercanviar-los per trajectes.
Prioritat	Mitja
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament) CU-0[3-6] (haver completat el procés d'embarcament)
Flux	L'usuari accedeix al menú dins la secció principal (*agenda)* i fa clic a l'opció de comprar crèdits. A continuació omple els seus detalls de la targeta de crèdit i realitza la compra.
Postcondicions	• El total de crèdits adquirits s'afegeix al balanç previ.

CU-17	Intercanviar crèdits per diners
Descripció	Un usuari pot retirar crèdits del seu compte i convertir-los en diners.
Prioritat	Mitja
Precondicions	CU-02 (haver accedit a l'aplicació satisfactòriament) CU-0[3-6] (haver completat el procés d'embarcament) L'usuari és conductor i té un mínim de 30 crèdits.
Flux	L'usuari accedeix al menú dins la secció principal (*agenda)* i fa clic a l'opció de retirar crèdits. A continuació omple els seus detalls de la targeta de crèdit i realitza la retirada.
Postcondicions	• El total de crèdits retirats es resta al balanç previ. • El nombre de crèdits retirats no serà superior al nombre de crèdits existents.

4.2. Arquitectura tècnica

4.2.1. Visió general

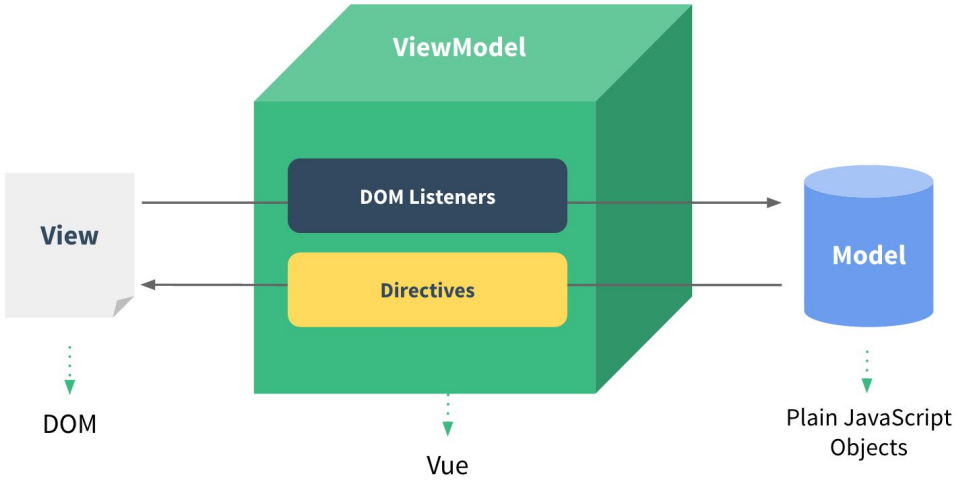


Vue	És un *framework* de *javascript* de codi obert el qual utilitza el model MVVM per crear interfícies d'usuari i aplicacions d'una sola pàgina (*SPA*). Creat per Evan You. [8]
Vuex	Es tracta d'una llibreria *javascript* de codi obert per la gestió d'estat específicament desenvolupada per complementar Vue. [9]
Service workers	Fortament impulsats per Google, un *service worker* és un *script* que s'executa de fons al navegador de manera independent respecte la pàgina web en qüestió, i el qual, introdueix funcionalitats que permeten a l'aplicació executar-se de la mateixa manera sense connexió a Internet (*offline*). [10]
Workbox	És una eina desenvolupada per Google la qual simplifica l'API per interactuar amb el Service Worker.
LocalForage	Es tracta d'una llibreria que simplifica l'accés a la memòria cau local (*Local Storage*).
Firebase	És un conjunt de productes i/o serveis oferits per Google especialment dirigits a facilitar i agilitzar el desenvolupament d'aplicacions progressives (PWA). [11]
Node.js	Es tracta d'una plataforma de codi obert i multiplataforma, la qual permet executar codi Javascript en un context diferent

	del d'un navegador web. [12]
Koa.js	És un *framework* minimalista per a Node.js, el qual permet desenvolupar APIs de manera senzilla.
MongoDB	És una base de dades la qual emmagatzema la informació en documents en format JSON i utilitza un llenguatge de consulta propi diferent de SQL (és a dir, NoSQL). [13]

4.2.2. Vue

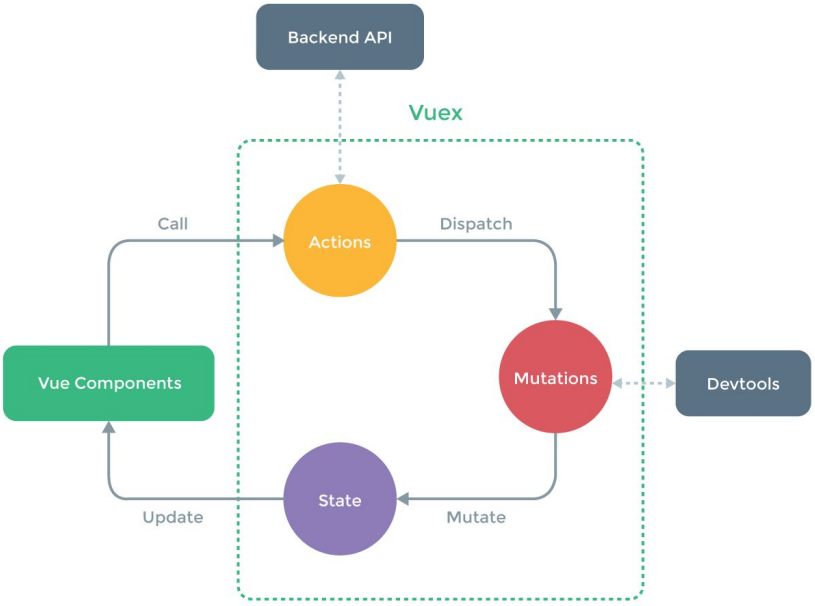
La llibreria *Vue* implementa la part *ViewModel* dins el paradigma *Model - View - ViewModel* (MVVM). Aquest paradigma facilita la separació del desenvolupament d'interfícies gràfiques. El *ViewModel* s'encarrega d'exposar els objectes de dades provinents del model de manera que siguin fàcils de gestionar i, normalment, encapsula la lògica de presentació.



4.2.3. Vuex

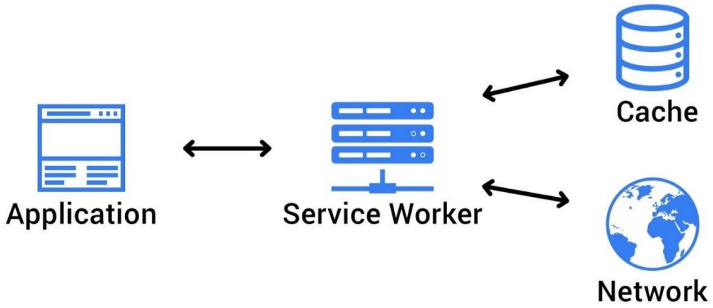
Vuex és una llibreria complementària de gestió d'estat unidireccional. L'estat de l'aplicació s'emmagatzema a l'abstracció **State**, el qual sol ser un objecte simple ("*plain object" a javascript*). A continuació es defineixen accions (**actions**), les quals interactuen amb l'API del servidor i s'encarreguen d'intercanviar informació i "envia" (*dispatch*) mutacions (*mutations*), les quals realitzen les modificacions a l'estat. La llibreria promou el paradigma d'immutabilitat, però depèn de l'usuari. Per tant, l'aplicació mai modifica l'estat directament, sinó que sempre segueix el flux descrit anteriorment, és a dir,

crida una **acció**, aquesta envia una o més **mutacions**, dins les quals es modifiquen les propietats de l'estat desitjades.



4.2.3. Service worker

Es tracta d'una eina relativament nova i promocionada especialment per Google. El seu ús primordial és per al desenvolupament d'aplicacions mòbils amb HTML5. Proporcionen principalment la capacitat que una aplicació funcioni sense connexió a internet de la mateixa manera (amb dades prèviament emmagatzemades, òbviament). El seu funcionament consisteix a interceptar les peticions al servidor i permetre definir una sèrie d'accions en resposta a aquestes peticions, per tant, funciona de manera similar a un *proxy.*



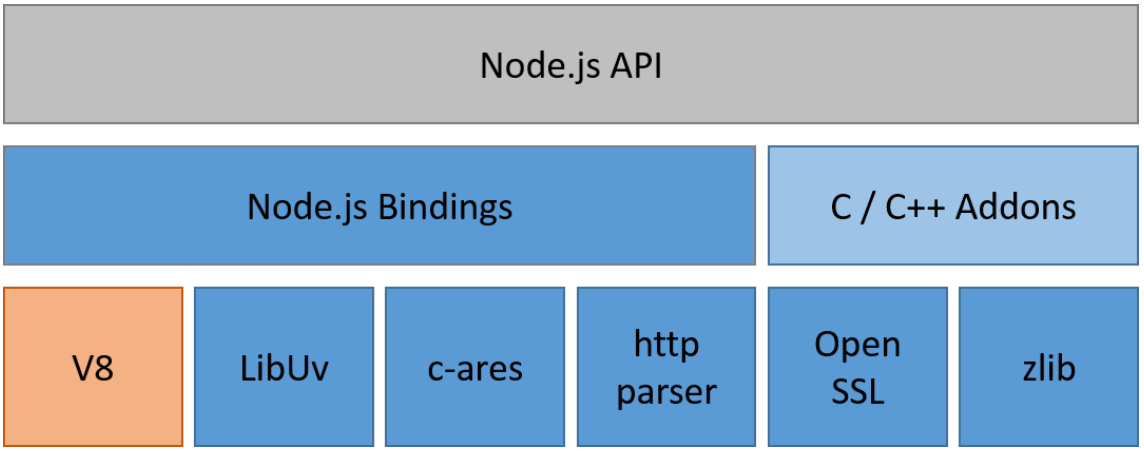
4.2.4. Firebase

Firebase és una gamma de productes i serveis oferits per Google, els quals faciliten i agilitzen el desenvolupament d'aplicacions web, i especialment, aplicacions progressives (PWA). A continuació es resumeixen els principals serveis:

Cloud Firestore	Base de dades NoSQL.
Cloud Functions	Funcions de crida independent (equivalent a les *Lambda Functions d'Amazon Web Services*).
Authentication	Servei d'autenticació que implementa mètodes aliens al propi Google, com Facebook o Twitter.
Hosting	Servei d'allotjament web amb accés a CDN (*content delivery network*) i SSL.
Cloud Storage	Servei d'emmagatzematge d'objectes (equivalent a *S3 d'Amazon Web Services*).
Realtime database	Base de dades de sincronització en temps real. Aquest producte és el "Firebase" original, el qual va ser adquirit per Google.
Cloud Messaging	Servei que permet enviar notificacions *push* a través d'un *service worker*.

4.2.5. Node.js

Node.js és una rutina desenvolupada sobre el motor javascript *Chrome V8* de codi obert. Aquest motor està programat en C++ i el que fa és transformar el codi javascript en codi de màquina i executar-ho. D'aquesta manera s'aconsegueix un gran rendiment. A part del V8, s'utilitzen altres llibreries i mòduls que componen els diferents serveis i mètodes que ofereix la seva API.



[14]

4.2.6. mongoDB

Avui en dia mongoDB ja no és només una base de dades, sinó que ha evolucionat per afegir tota una gamma de productes i serveis els quals complementen i faciliten l'ús de mongoDB com a base de dades dins un projecte. Les principals característiques de mongoDB són les següents:

- **Documents JSON**: les dades s'emmagatzemen com a documents independents en format JSON amb suport per esquemes dinàmics.
- **Llenguatge de consulta propi**: és una base de dades categoritzada com a NoSQL, i té un llenguatge de consulta expressiu específicament desenvolupat per consultar documents. Les consultes són també en format JSON i són, per tant, combinables, permetent una gran flexibilitat a l'hora de realitzar consultes complexes.
- **Altres:**
    - Suport per transaccions ACID.
    - Suport per *JOINS*.

Més enllà de la base dades, cal destacar el servei *Atlas*, el qual ofereix una base de dades al núvol, i utilitzarem en aquest projecte.

4.3. Disseny de dades

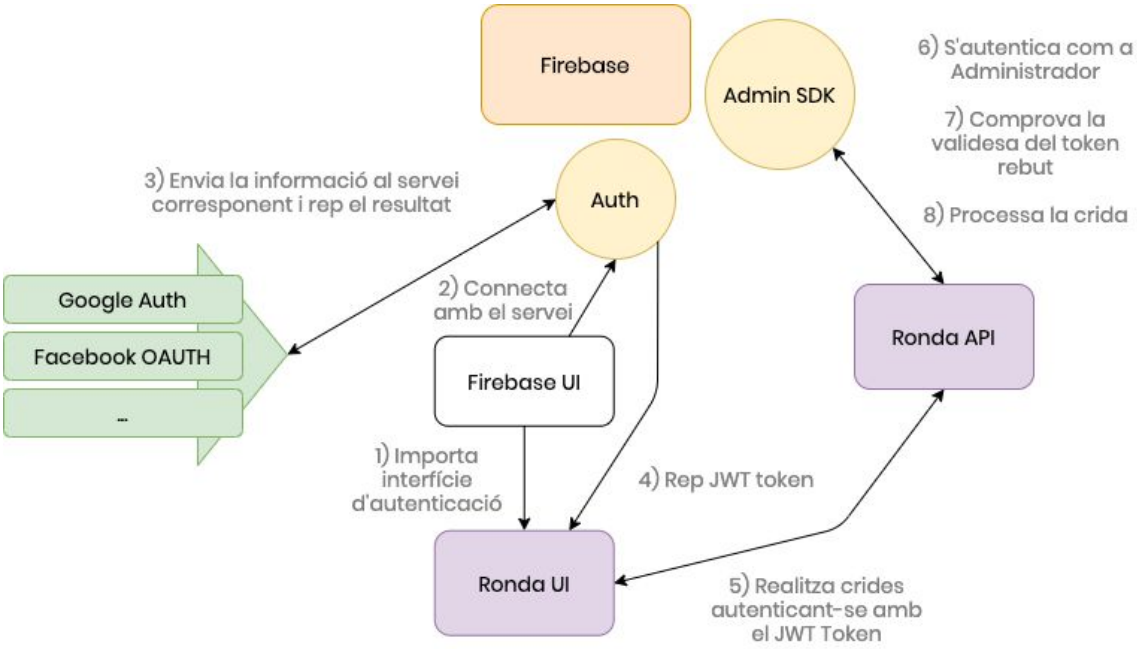
5. Implementació

A continuació es detallen diverses implementacions seleccionades per il·lustrar les tecnologies i funcionalitats més rellevants. Per a cada implementació s'aporta una breu descripció, un diagrama explicatiu i parts del codi font comentades. Les eines utilitzades pel desenvolupament són, d'una banda, l'editor Visual Studio Code de Microsoft per escriure el codi font, d'altra banda el navegador Google Chrome, amb el qual s'executa i inspecciona l'aplicació.

5.1. Autenticació amb Firebase

5.1.1. Visió general

Firebase ens proporciona un servei d'autenticació que permet a l'usuari utilitzar els seus comptes existents d'altres serveis, com Facebook, Twitter o Gmail, per autenticar-se en la nostra aplicació. A continuació es detalla l'arquitectura i s'explica el procés d'implementació:



Tot i que no és un requisit indispensable, és molt recomanable utilitzar la llibreria Firebase UI [15] (en el nostre cas, la versió web). Es tracta d'una llibreria amb components d'interfície per a tot el flux del procés d'autenticació, per tant, ens estalvia molta feina. Quan un usuari s'autentica a través d'un dels mecanismes disponibles, l'aplicació rep un JWT (JSON Web Token) [16] amb el qual s'han d'autoritzar les crides a l'API.

Dins l'API, primer s'ha d'autenticar com a administrador autoritzat pel projecte en qüestió a través d'una clau específica, i a continuació, es realitza una crida al servei per tal de validar el *JWT*. D'aquesta manera la nostra API no ha de gestionar cap aspecte d'autenticació, més enllà de realitzar les crides corresponents a Firebase.

A continuació s'inclouen les parts de codi principals responsables d'aquesta funcionalitat.

5.1.2. Implementació

Implementació de la interfície:

Primer de tot importem les llibreries necessàries:

```
import Vue from "vue";
import firebase from "firebase/app";
import * as firebaseui from "firebaseui";
```

Creem un component Vue anomenat **ron-auth** que gestiona el procés d'autenticació. Quan aquest component es crea (és a dir, s'accedeix a la pàgina d'autenticació) s'afegeix una font d'estils CSS específica per a Firebase UI.

```
export default Vue.extend({
 name: "ron-auth",
 created() {
   const head = document.getElementsByTagName("head")[0];
   const link = document.createElement("link");
   link.rel = "stylesheet";
   link.type = "text/css";
   link.href =
"https://www.gstatic.com/firebasejs/ui/4.3.0/firebase-ui-auth.
css";
   link.media = "all";
   head.appendChild(link);
 },
```

Un cop el component es munta (és a dir, l'HTML corresponent s'ha afegit al DOM), es crida a la funció **getInstance** de la llibreria d'autenticació i s'envia l'element **ron-auth-container** com a contenidor de la interfície, i els paràmetres de configuració necessaris. En aquest s'ha optat per permetre l'autenticació a través d'e-mail i telèfon.

```
 mounted() {
   const uiConfig = {
     signInSuccessUrl: "/",
     signInOptions: [
```

```
        firebase.auth.EmailAuthProvider.PROVIDER_ID,
        firebase.auth.PhoneAuthProvider.PROVIDER_ID
      ],
      tosUrl: "tos-url", // todo
      // Privacy policy url/callback.
      privacyPolicyUrl: "privary-url" // todo
    };
    const ui =
      firebaseui.auth.AuthUI.getInstance() ||
      new firebaseui.auth.AuthUI(firebase.auth());
    ui.start("#ron-auth-container", uiConfig);
  }
});
```

Implementació del backend:

firebase.ts
Creem un mòdul específic per realitzar l'autenticació com a administrador del projecte.

A l'arxiu **key.json** tenim tota la informació relacionada amb la configuració i la nostra clau secreta.

Aquesta informació s'envia al mètode **initializeApp** i l'autenticació queda realitzada.

A continuació s'exporta la funció **verify**, a través de la qual es realitzarà la verificació dels *tokens* JWT que autoritzen les crida des de la interfície. En aquesta funció simplement es passa el *token* a la funció **verifyIdToken** de l' *admin SDK* i es rep descodificat.

```
import * as admin from "firebase-admin";

const serviceAccount = require("./firebase/key.json");

admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
});

export default async function verify(idToken: string) {
  try {
    const decodedToken = await
admin.auth().verifyIdToken(idToken);
    return decodedToken;
  } catch (err) {
    console.error("Token verification", err);
    throw err;
```

```
  }
}
```

authorize.ts
Per tal d'autoritzar les crides des de la interfície, creem un *middleware* pel qual hi passaran totes.

Extraiem el *token* de la crida llegint la capçalera *fb-id-token* i cridem el mètode **verify** que hem creat anteriorment. A continuació, un cop tenim la informació que conté el *token*, busquem un usuari a la base de dades que concordi amb l'identificador. Si en trobem un s'autoritza la crida, en cas contrari es retorna amb estat HTTP 401, "crida no autoritzada".

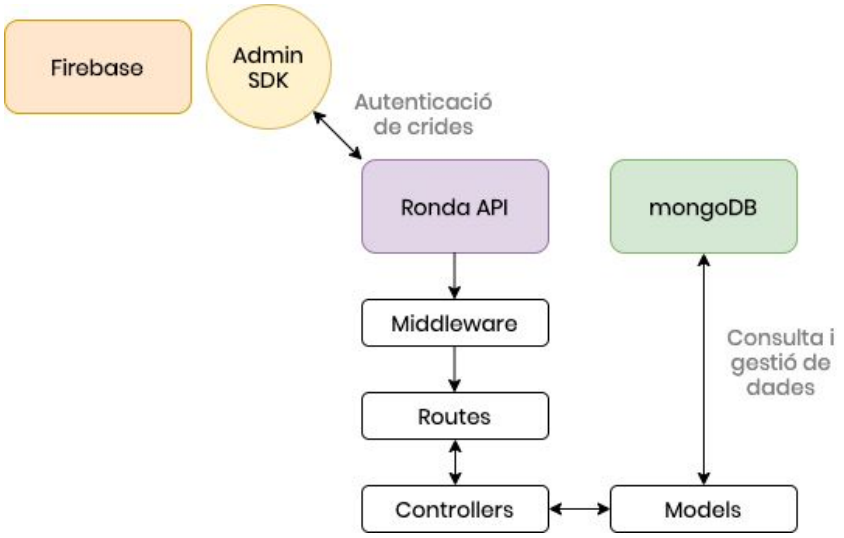
```
export default async function(
  ctx: Koa.ParameterizedContext<Koa.DefaultState,
Koa.DefaultContext>,
  next: Koa.Next
) {
  try {
    const idToken = ctx.header["fb-id-token"];
    if (!idToken) {
      throw Error("Could not find the fb-id-token header in
the request.");
    }
    const decodedToken = await verify(idToken);
    const account = await Account.findOne({ user:
decodedToken.uid });
    if (!account) {
      throw Error("Could not find an account for the fb user
id.");
    }
    ctx.state.user = { accountId: account._id, fbId:
account.user };
  } catch (e) {
    ctx.status = 401;
    ctx.body = "Unauthorized request.";
  }
  return next();
}
```

server.ts
Finalment, al servidor simplement hem de passar el *middleware* al servidor a través del mètode **use**.

```
import Koa from "koa";
import { authorize } from "./middleware";
const app = new Koa();
app.use(authorize);
```

5.2. REST API amb Koa.js

5.2.1. Visió general



Per implementar l'API s'ha utilitzat el *framework* Koa.js [17], el qual es defineix com a compacte, expressiu i robust. L'arquitectura de l'API segueix el model MVC (*model-view-controller*), tot i que no implementa la vista, ja que les crides retornen exclusivament dades en format JSON. A més, s'organitza en 4 àrees principals:

- **Account**: fa referència a la gestió de comptes d'usuari.
- **Booking**: aporta la funcionalitat necessària per gestionar les reserves.
- **Network**: inclou mètodes per gestionar l'àrea de *networking*.
- **Trip**: tracta la disponibilitat dels usuaris ("trajectes").

Es defineixen les següents rutes:

Mètode	Ruta	Descripció
GET	/account/self	Retorna el compte de l'usuari qui realitza la crida.
GET	/account/:*id*	Retorna el compte de l'usuari amb la id especificada.
PUT	/account	Modifica el compte de l'usuari qui realitza la crida.
GET	/bookings	Retorna les reserves de l'usuari qui realitza la crida.

POST	/bookings	Crea una nova reserva.
PUT	/bookings/update-status	Actualitza l'estat d'una reserva.
GET	/network/candidates	Retorna els candidats per a l'usuari que realitza la crida.
POST	/network/swipe	Crea una decisió sobre un candidat.
GET	/network/matches	Retorna les coincidències per a l'usuari que realitza la crida.
PUT	/network/:swipeId	Canvia la decisió sobre un candidat.
GET	/trips/self	Retorna la disponibilitat de l'usuari que realitza la crida.
GET	/trips/all	Retorna els trajectes compatibles dins la base d'usuaris de la plataforma.
PUT	/trips/update/*:id*	Modifica un trajecte amb l'identificador corresponent.

5.2.2. Implementació

routes/booking.ts

Es defineixen les rutes referents a reserves. S'utilitza la llibreria **koa-router**, la qual exporta una classe **Router** que ens permet crear l'encaminador. Definim el prefix **/bookings**, el qual utilitzarem per a totes les rutes definides en aquest arxiu. Es defineix la ruta **GET /**, on es crida al controlador **BookingController** per tal d'obtenir les reserves de la base de dades.

```
import Router from "koa-router";
import BookingController from "../controllers/Booking";
import { IBooking, EnumBookingStatus } from "../types";

const router = new Router({
  prefix: "/bookings"
});

router.get("/", async (ctx, next) => {
  try {
    const { accountId } = ctx.state.user;
    const docs = await BookingController.findSelf(accountId);
    ctx.body = docs;
  } catch (err) {
    ctx.status = 500;
```

```
  }
  next();
});
```

controllers/Booking.ts

El patró típic pel controlador és el de cridar al model per obtenir les dades necessàries i retornar-les. Tot i això, en la majoria de casos es requereix afegir un altre nivell de lògica o realitzar altres operacions.

```
export default {
  find: async (id: string) => {
    try {
      const doc = await BookingModel.findOne({ id });
      return doc;
    } catch (err) {
      console.error(err);
      throw err;
    }
  }
// [...]
}
```

models/Booking.ts

Per a la definició dels models s'utilitza la llibreria **Mongoose**, la qual ens permet definir els esquemes i realitzar les validacions automàticament. Per definir un camp de dades, simplement s'ha d'especificar el tipus de dades (*type*). Altres opcions són el paràmetre *required*, el qual defineix el camp com a obligatori, i el paràmetre *ref*, el qual estableix que el camp és una relació amb un altre model.

```
import { Schema, model } from "mongoose";
import { IBooking, IBookingModel } from "../types";

const { ObjectId } = Schema.Types;

const schema = new Schema({
  tripId: {
    type: ObjectId,
    required: true,
    ref: "Trip"
  },
  date: {
    type: Date,
    required: true
  },
  type: {
    type: String,
    required: true
```

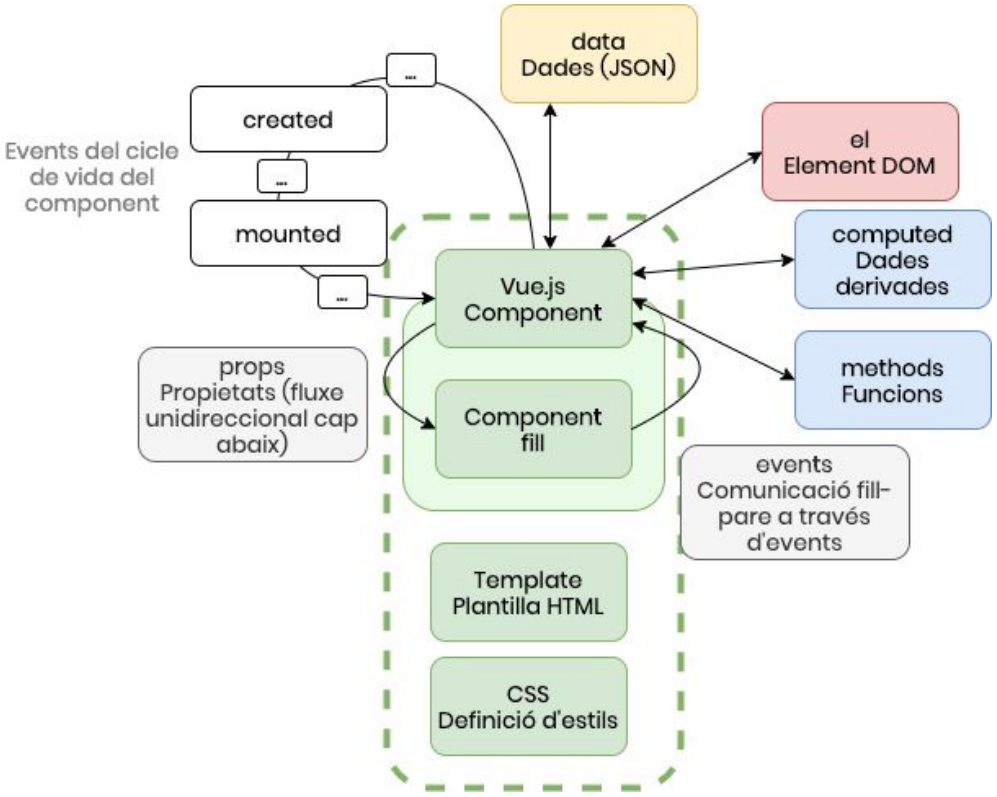
```
  },
// [...]

export default BookingModel;
```

5.3. Components mòbils amb Vue.js

5.3.1. Visió general

Per tal d'implementar la interfície s'ha utilitzat únicament el *framework* Vue.js. Tot i que existeixen algunes llibreries de components per a mòbils com *Ionic*, la necessitat de diversos components específics i també per preferència personal, s'ha optat per una implementació des de zero. A continuació es mostra un diagrama amb l'estructura bàsica d'un component a Vue:



Una de les característiques diferencials quant a interfície en els dispositius mòbils és la capacitat de navegar a través de gestos, i especialment, el canvi de pantalla a través del desplaçament, típicament anomenat *Swipe*. Per tal de permetre aquest tipus de navegació, s'ha creat un component de tipus contenidor, és a dir, un component que conté altres components, amb l'objectiu que sigui reutilitzable. A continuació es mostra la implementació:

components/swiper.ts

La plantilla consta simplement de l'element contenidor, sobre el qual es registren els esdeveniments corresponent als gests *touchstart, touchmove, touchend*, i els mateixos esdeveniments pel ratolí (en cas que s'utilitzi l'aplicació des d'un ordinador). Aquests esdeveniments venen incorporats per Vue i s'ha utilitzat un paquet específic pels esdeveniments tàctils *vue2-touch-events*.

```
<template>
  <div
    class="ron-swiper"
    v-on:mousedown="dragStart"
    v-on:touchstart="dragStart"
    v-on:mousemove="drag"
    v-on:touchmove="drag"
    v-on:mouseup="dragEnd"
    v-on:touchend="dragEnd"
  >
    <slot></slot>
  </div>
</template>
```

El component accepta 2 paràmetres (*props*), *screens* i *current*, els quals fan referència al nombre de pantalles i l'identificador de la pantalla actual.

Un cop el component s'ha muntat al *DOM*, fixem l'amplada, la qual ve determinada pel nombre de pantalles que conté i es fixa la posició inicial (la pantalla definida per la propietat *current*).

```
import Vue from "vue";
export default Vue.extend({
  name: "ron-swiper",
  props: ["screens", "current"],
  mounted() {
    this.target = document.querySelector(".ron-swiper");
    this.target.style.width = `${this.screens * 100}%`;
    this.target.style.left = `${this.currentLeft}px`;
  }

// [...]
computed: {
    currentIndex: function() {
```

```
      return this.current || 0;
    },
    currentLeft: function() {
      return -(this.fullWidth * this.currentIndex);
    }
  },
```

A continuació es defineixen els mètodes corresponents a cada esdeveniment (*event handlers*). Quan comença el desplaçament simplement es defineix la variable *isDragging* com a vertadera, la qual defineix l'estat.

Quan s'està produint el desplaçament (el mètode *drag*), es canvia la posició de l'element per seguir el dit. La posició és definida per l'esdeveniment (*e.touches[0].clientX*).
A més, s'ha definit un límit (*boundary*) el qual un cop es passa, es canvia l'opacitat de l'element, indicant així que si s'acaba el desplaçament en aquesta posició, es produirà el canvi de pantalla.

```
dragStart: function(e) {
  // [...]
  this.isDragging = true;
}

drag: function(e) {
    if (this.isDragging) {
      const { boundary, preventDragging, threshold } =
this.settings;
      if (e.type === "touchmove") {
        this.currentX = e.touches[0].clientX - this.startX;
      } else {
        this.currentX = e.clientX - this.startX;       }
      if (!preventDragging || Math.abs(this.currentX) >
threshold) {
        this.settings.preventDragging = false;
        const left = this.currentLeft + this.currentX;
        this.target.style.left = `${left}px`;
        if (this.currentX < -boundary) {
          this.target.style.opacity = 0.5;
        } else if (this.currentX > boundary) {
          this.target.style.opacity = 0.5;
        } else {
          this.target.style.opacity = 1;
        }
      }
    }
  },
```

Un cop s'acaba el desplaçament, es torna a canviar l'opacitat al valor normal (1). Es defineix si el desplaçament s'ha produït cap a l'esquerra o cap a la dreta, en funció de la posició final. Si el desplaçament ha superat el límit, llavors es realitza l'acció,

en cas contrari es cancel·la el desplaçament i es torna a la posició inicial.

```
dragEnd: function(e) {
      this.target.style.opacity = 1;
      const { boundary, animationTime } = this.settings;
      if (this.isDragging) {
        this.target.style.transition = `left
${animationTime}s`;
        if (this.currentX <= -boundary) {
          // left (forward)
          this.handleSwipe(true);
        } else if (this.currentX >= boundary) {
          // right (backward)
          this.handleSwipe(false);
        } else {
          this.cancelSwipe();
        }
        this.isDragging = false;
        this.currentX = 0;
        this.currentY = 0;
        this.settings.preventDragging = true;
      }
    },
```

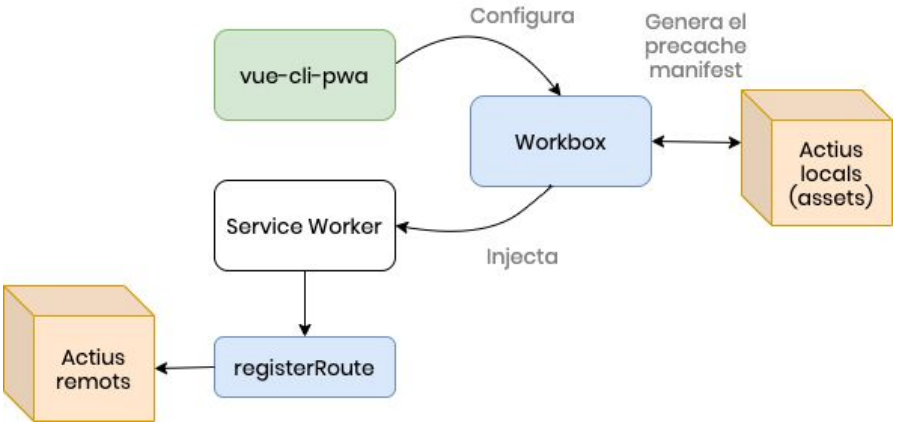
Finalment es realitza el desplaçament canviant el valor *left* de l'element al corresponent de la pantalla. Abans es comprova si es tracta d'un desplaçament a l'esquerra a la pantalla inicial, o pel contrari, un desplaçament a la dreta a l'última pantalla; en ambdós casos es cancel·la tornant a la posició inicial.

```
handleSwipe(bool) {
  const direction = bool ? "left" : "right";
  if (
    (bool && this.currentIndex === this.screens - 1) ||
    (!bool && this.currentIndex === 0)
  ) {
    this.cancelSwipe();
  } else {
    this.$emit("swipe", direction);
    Vue.nextTick(() => (this.target.style.left =
`${this.currentLeft}px`));
  }
}
```

5.4. Suport *offline* amb Workbox i LocalForage

5.4.1. Visió general

Hi ha diverses situacions on els dispositius mòbils poden perdre la connectivitat, per la qual cosa, és important que una aplicació pugui funcionar *offline*, encara que sigui de manera limitada. Per a aconseguir-ho s'ha utilitzat l'eina *Workbox*, la qual facilitat la definició del *Service Worker*, així com la llibreria *LocalForage*, que ens permet gestionar l'emmagatzematge local de manera senzilla.



Workbox

vue.config.js
Per carregar *workbox* utilitzem el plug-in **vue-cli-pwa** i passem les opcions de configuració següents:

- mode: *InjectManifest*, injecta workbox a l'arxiu que definim, en lloc de crear un arxiu de manera automàtica, això ens permet més opcions de configuració.
- swSrc: *public/service-worker.js*, defineix l'arxiu on s'injecta workbox.

```
pwa: {
    // configure the workbox plugin
    workboxPluginMode: 'InjectManifest',
    workboxOptions: {
      // swSrc is required in InjectManifest mode.
      swSrc: 'public/service-worker.js',
      // ...other Workbox options...
    },
    themeColor: '#3443A2'
  }
```

public/service-worker.js
Tenim la variable *__precacheManifest*, la qual és un arxiu creat automàticament pel plug-in vist anteriorment i conté tots els arxius que hem afegit a la *build* (imatges, estils, fonts, etc.).
El mètode *precacheAndRoute* introdueix els actius especificats a la memòria cau abans que la pàgina es carregui. He hagut de passar la funció *urlManipulation* per tal d'extreure els paràmetres de les URLs)

```
self.__precacheManifest = [].concat(self.__precacheManifest ||
[]);
workbox.precaching.suppressWarnings();
workbox.precaching.precacheAndRoute(self.__precacheManifest, {
  urlManipulation: ({url}) => {
    // Remove query params from urls
    const alteredUrl = Object.assign({}, url, { href:
url.href.replace(/\?[0-9a-zA-Z\=\.\-]+/g, "") });
    return alteredUrl;
  }
});
```

Per aquells actius que no disposem, workbox ens proporciona un mòdul *routing* amb el mètode *registerRoute*, amb el qual podem introduir a la memòria cau aquests actius remots. A part de registrar l'URL, aquí podem triar una estratègia a l'hora de carregar l'arxiu les pròximes vegades. Les opcions principals són utilitzar la memòria cau primer i anar a l'URL si no es disposa de l'arxiu o es vol actualitzar sempre, o a l'inrevés, anar sempre a buscar el recurs a la xarxa. En aquest cas utilitzem *StaleWhileRevalidate*, que és la primera opció descrita.

```
// Cache the Google Fonts stylesheets with a
stale-while-revalidate strategy.
workbox.routing.registerRoute(
  /^https:\/\/fonts\.googleapis\.com/,
  new workbox.strategies.StaleWhileRevalidate({
    cacheName: 'google-fonts-stylesheets',
  })
);
```

LocalForage

localforage.ts

Primer configurem la llibreria definint una sèrie d'opcions com la mida i el *driver* utilitzat, i després creem una instància a la qual anomenem *ronda-store*.

```
import localforage from "localforage";

localforage.config({
  driver: localforage.WEBSQL,
  name: "ronda",
  version: 1.0,
```

```
  size: 1000000 // bytes
});

const localStore = localforage.createInstance({
  name: "ronda-store"
});

export default localStore;
```

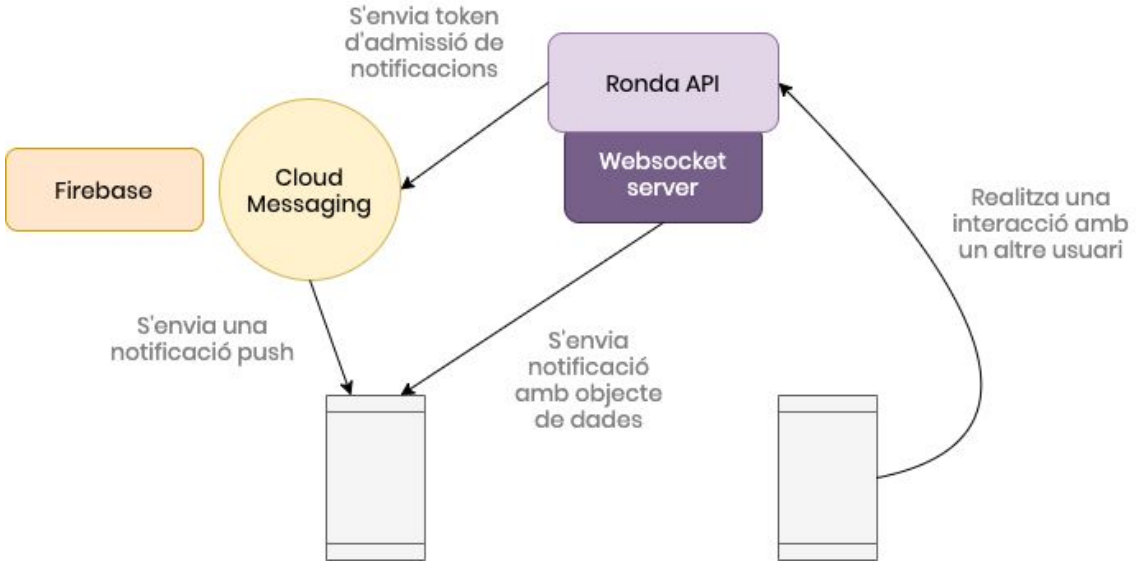
A continuació es demostra l'ús de la llibreria en una acció de *Vuex*. En aquest cas l'acció demana la informació de l'usuari autenticat a l'API. Comprovem si el dispositiu està en línia i obtenim les dades, en aquest punt cridem el mètode *setItem* i guardem les dades rebudes a la ruta definida pel recurs. Per contra, si estem en mode *offline*, recuperem les dades locals a través del mètode *getItem*.

```
async fetchAccount({ commit }: { commit: Function }) {
  const resource = "account/self";
  let account;
  if (navigator.onLine) {
    try {
      const response = await api.get(resource);
      account = response.data;
      await localStore.setItem(resource, account);
    } catch (err) {
      console.log(err);
    }
  } else {
    account = await localStore.getItem(resource);
  }
  commit("setAccount", { account });
}
```

5.5. Notificacions amb Firebase i Websockets

5.5.1. Visió general

Les notificacions són un element fonamental de les aplicacions mòbils. En el cas de les aplicacions progressives, les notificacions s'implementen a través del *service worker*, i es coneixen com a notificacions *push*. Firebase ens ofereix un servei per facilitar l'enviament i recepció d'aquestes. Tot i això, aquestes notificacions tenen una limitació quant al contingut que es pot enviar al dispositiu des del servidor, per la qual cosa, he desenvolupat un sistema de notificacions paral·lel amb la tecnologia *websockets*. Es distingeixen dos casos principals, un quan l'usuari té l'aplicació oberta (*in-app notifications* amb *websockets*), i un segon quan l'usuari té l'aplicació tancada (*push notifications* amb *Firebase*).



5.5.2. Implementació

App.vue
Dins el component principal de l'aplicació definim una funció per demanar permís a l'usuari per enviar-li notificacions. Utilitzem el mòdul *messaging* de Firebase i comprovem si hi ha tenim permís. En cas negatiu es demana l'acceptació de l'usuari i s'emmagatzema el *token* obtingut a la base de dades. Aquest *token* s'ha d'afegir a totes les crides de notificacions que s'envien des de l'API.

```
async askForNotifications() {
  try {
    const messaging = firebase.messaging();
```

```
    let token = await messaging.getToken();
    console.log("current push notifications token:", token);
    if (!token) {
      await messaging.requestPermission();
      token = await messaging.getToken();
    }
    await this.changeValue({
      key: "metadata",
      id: "pushNotificationsToken",
      value: token,
    });
  //…
}
```

notifications/index.ts
Al servidor hem creat un mòdul específic amb diverses funcions per realitzar les tasques relacionades amb les notificacions. A continuació es mostra el codi de la funció **push**, la qual envia la informació a l'API de Firebase Cloud Messaging (FCM). Per realitzar aquestes crides, a part del *token* vist anteriorment (paràmetre *to* a FCM), hem d'autoritzar el client amb una clau, *messaging_server_key*, la qual podem obtenir a la consola del nostre projecte de Firebase.

```
async function push(notification: IPushNotification, to:
string) {
  try {
    await axios({
      url: "https://fcm.googleapis.com/fcm/send",
      method: "post",
      headers: {
        "Content-Type": "application/json",
        Authorization: `key=${fbConfig.messaging_server_key}`
      },
      data: { notification, to }
    });
  } catch (err) {
    console.error(err);
  }
}
```

notification/templates.ts
Les dades concretes que s'envien dins les notificacions es troben a aquest arxiu, el qual reuneix totes les plantilles disponibles per cada acció que es vol notificar. S'han definit com a funcions que accepten una sèrie de valors, els quals són substituïts a l'objecte a retornar.

```
//…
newBookingRequest: function(values: { firstName: string }) {
    const { firstName } = values;
    return {
      title: "Nova sol·licitud 🚘",
      body: `${firstName} tens un nou passatger interessat en
```

```
acompanyar-te!`,
      click_action: "http://localhost:8080/",
      icon: "http://localhost:8080/icon.png"
    };
  }
 //…
```

socket.ts
He definit una classe *Socket*, la qual s'encarrega de connectar amb el servidor de websockets i gestionar els missatges rebuts. Cal destacar l'ús d'una instància de Vue a manera de bus global d'esdeveniments, és a dir, quan es rep un missatge a través de websockets, s'emet un esdeveniment a una instància de Vue, la qual hem anomenat *bus* i pot ser importada (per tant escoltar els esdeveniments) per qualsevol component de la nostra aplicació.

```
class Socket {
  ws: WebSocket | null = null;
  token: string = "";
  url: string = process.env.VUE_APP_WS ||
"ws://localhost:9090";
  reconnectAttempts: number = 0;
  reconnectLimit: number = 12;
  reconnectDelay: number = 5000;

  connect(token: string) {
   //…
  }

 handleMessage(message: MessageEvent) {
    const { data } = message;
    try {
      const payload = JSON.parse(data);
      const { type, content } = payload;
      console.log("Socket: emitting", type, content);
      bus.$emit(`socket-${type}`, content);
    } catch (err) {
      console.error("Could not parse socket message,", data);
    }
 }
}
```

bus.ts
Com s'ha comentat anteriorment el bus és simplement una instància de Vue. He definit un *helper* anomenat *listenTo*, el qual registra els components que escolten un tipus d'esdeveniments i enregistren una *callback* per tal que s'executi quan aquests esdeveniments es produeixen.

```
const bus = new Vue({});
const listening: Array<{ component: string; listeners:
Array<String> }> = [];
```

```
export function listenTo(eventName: string, fn: Function,
component: string) {
  const record = listening.find(item => item.component ===
component);
  const isRegistered = record &&
record.listeners.includes(eventName);
  if (isRegistered) {
    return;
  }
  bus.$on(eventName, fn);
  if (record) {
    record.listeners.push(eventName);
  } else {
    listening.push({ component, listeners: [eventName] });
  }
}
```

socket.ts
A l'API hem definit també un mòdul *socket* per tal de gestionar la creació del servidor i gestionar la connexió dels clients. A més inclou un mètode *send*, a través del qual s'envien les notificacions als usuaris. A diferència de les notificacions *push*, a través de websockets podem enviar els objectes pels quals es realitza la notificació, per tal que l'usuari rebi la informació al mateix temps que la notificació. Per exemple, si hi ha una nova notificació, aquesta s'enviaria dins l'argument *content*, i un cop arriba al dispositiu, l'aplicació l'afegiria directament a les dades sense haver de realitzar una crida addicional al servidor.

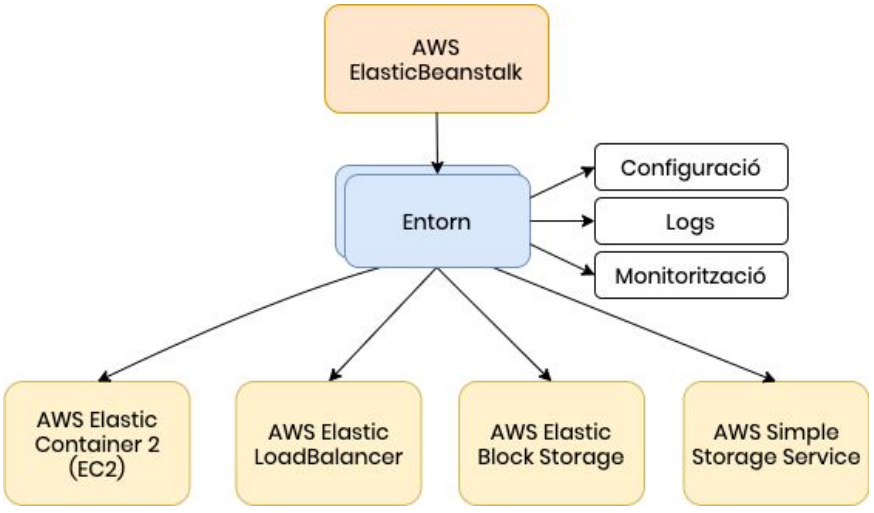
```
class SocketManager {
  static send(accountId: string, type: string, content: any) {
    const client = clients[accountId];
    if (typeof client === "undefined") {
      console.error("SocketManager: no client with user id",
accountId, type);
      return;
    }
    console.log("Sending:", type, accountId);
    client.ws.send(JSON.stringify({ type, content }));
  }
}
```

5.6. Desplegament del Backend amb AWS Elastic Beanstalk

5.6.1. Visió general

Per tal de desplegar el *backend* he triat Amazon Web Services, ja que és un dels proveïdors més fiables, amb més serveis i escalable automàticament, i a més, ofereix una capa de serveis amb limitacions, de manera gratuïta.

Concretament he utilitzat *Elastic Beanstalk*, el qual crea un entorn per l'aplicació i dóna accés a una sèrie d'eines com la configuració, els registres i el monitoratge. Però el més important és que s'encarrega de crear automàticament els serveis necessaris per fer funcionar l'aplicació, descrits a continuació:



Servei	Descripció
Elastic Container 2	La màquina on s'executa l'aplicació.
Elastic LoadBalancer	El balancejador de càrrega s'encarrega de distribuir el tràfic entrant de manera equitativa per tal que l'aplicació sigui escalable.
Elastic Block Storage	Emmagatzematge de tipus bloc on es troben els arxius de codi font.
Simple Storage Service	Emmagatzematge de tipus objecte on es troba la configuració.

5.6.2. Implementació

En primer lloc, haurem d'instal·lar el *Elastic Beanstalk Command-Line Interface (eb cli)*, el qual ens permetrà configurar el desplegament amb unes poques comandes, concretament: **eb init** i **eb deploy**. La primera comanda ens realitza una sèrie de preguntes per tal de configurar el projecte, com per exemple la zona on volem que es desplegui. Un cop configurat, realitzem un *eb deploy* i comprovem a la interfície d'*AWS* que el projecte s'ha desplegat. Automàticament obtenim un URL on s'executa la nostra aplicació.

La configuració es guarda dins el projecte al següent arxiu en format YAML:

.elasticbeanstalk/config.yml
Hem anomenat l'entorn *api-staging*, ja que l'entorn final de producció es configurarà més endavant. La nostra plataforma és *Node.js* i la zona seleccionada és *eu-north-1*, la qual correspon a Estocolm (Suècia). S'utilitza GIT com a programari de control de versions.

```
branch-defaults:
  master:
    environment: api-staging
global:
  application_name: api
  default_platform: Node.js
  default_region: eu-north-1
  include_git_submodules: true
  sc: git
  workspace_type: Application
```

A més, per aquesta aplicació hem hagut d'afegir diverses configuracions addicionals. El *eb cli* ens permet afegir aquestes configuracions dins un directori **.ebextensions** en tants arxius com desitgem, amb l'extensió *.config*. A continuació es detallen les extensions utilitzades:

.ebextensions/source_compile.config
L'API s'ha programat amb *TypeScript*, el qual s'ha de compilar per obtenir el codi javascript a executar. A través d'aquesta extensió s'indica la realització d'aquest pas previ.

```
container_commands:
  compile:
    command: "./node_modules/.bin/tsc -p tsconfig.json"
    env:
      PATH:
/opt/elasticbeanstalk/node-install/node-v10.15.3-linux-x64/bin
/
```

.ebextensions/nodecommand.config
A través d'aquesta extensió definim la comanda que cal introduir per executar l'aplicació, *npm start*. A més, afegim un paràmetre per tal que *npm* (Node Package Moduler) instal·li les dependències de desenvolupament (*devDependencies*), ja que *TypeScript* n'és una d'elles i és necessària per compilar el codi, tal com s'ha indicat anteriorment.

```
option_settings:
  aws:elasticbeanstalk:application:environment:
    NPM_USE_PRODUCTION: false
  aws:elasticbeanstalk:container:nodejs:
    NodeCommand: "npm start"
```

.ebextensions/https.config
A través d'aquesta extensió activem el funcionament de SSL (Socket Security Layer) per tal d'utilitzar el protocol d'HTTPS. Cal mencionar que s'ha d'obtenir un certificat amb AWS Certificate Manager i configurar l'AWS Load Balancer per tal que funciona correctament. A més, és un requisit obligatori, ja que el Service Worker només funciona a través d'aquest protocol, per qüestions de seguretat.

```
Resources:
  sslSecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: {"Fn::GetAtt" : ["AWSEBSecurityGroup",
"GroupId"]}
      IpProtocol: tcp
      ToPort: 443
      FromPort: 443
      CidrIp: 0.0.0.0/0
```

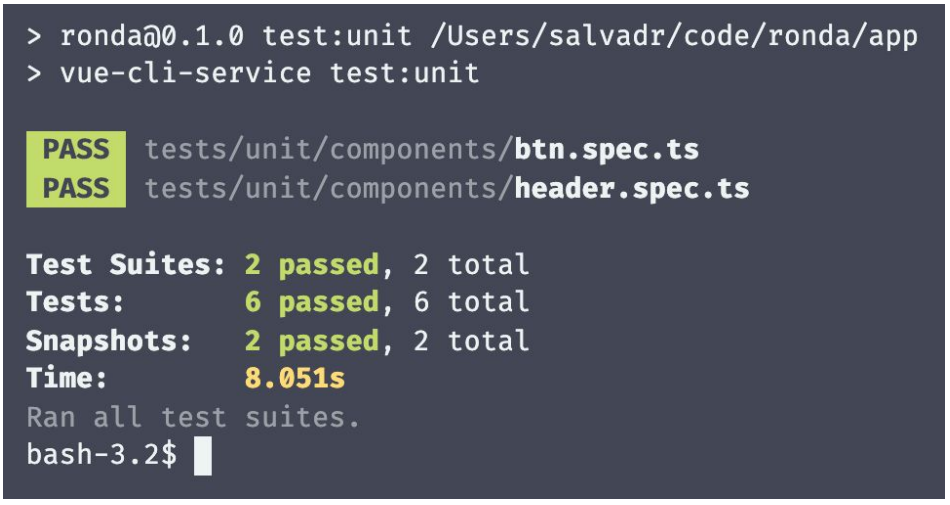
.ebextensions/websockets.config
Per habilitar el funcionament de Websockets hem d'afegir aquesta extensió o es configura el servidor *proxy* per tal que *promogui* les connexions d'HTTPS a WSS.

```
container_commands:
  enable_websockets:
    command: |
      sed -i '/\s*proxy_set_header\s*Connection/c \
              proxy_set_header Upgrade $http_upgrade;\
              proxy_set_header Connection "upgrade";\
              '
/tmp/deployment/config/#etc#nginx#conf.d#00_elastic_beanstalk_
proxy.conf
```

6. Proves

6.1. Proves unitàries

Per realitzar les proves unitàries utilitzarem el *framework Jest*[18], el qual disposa d'una integració amb *Vue CLI*. La codificació dels tests es troba a la carpeta /**tests/unit** i es poden executar amb la comanda **npm run test:unit**.



A continuació es mostra un exemple, on es comprova que el component disposa de certes propietats:

tests/unit/components/header.spec.ts

```
import { shallowMount } from "@vue/test-utils";
import header from "@/components/header/header.vue";

const wrapper = shallowMount(header);

describe("header", () => {
  it("is a valid component", () => {
    expect(wrapper.isVueInstance()).toBeTruthy();
  });
  it("matches the snapshot", () => {
    expect(wrapper.element).toMatchSnapshot();
  });
  it("has a goBack method", () => {
    expect((wrapper.vm as any).goBack).toBeTruthy();
  });
});
```

Per raons de temps no he pogut completar la suite de tests per a tots els components, es deixa com a tasca per a més endavant.

7. Revisió de la planificació

7.1. Entrega PAC3

L'estat del projecte és òptim en relació amb la planificació realitzada, l'alta familiaritat amb la majoria de les tecnologies utilitzades ha permès un desenvolupament fluid.

Tasca	Inici	Fi	Estat
Implementació de la interfície d'usuari (HTML5,CSS, VUE)	02/04/2020	11/04/2020	**Complert**
Desenvolupament de la lògica de client (VUE, VUEX)	12/04/2020	20/04/2020	**Complert**
Configuració de base de dades (MongoDB)	21/04/2020	21/04/2020	**Complert**
Desenvolupament API servidor (Node.js)	22/04/2020	01/05/2020	**Complert**
Integració de servei d'autenticació (Firebase)	02/05/2020	07/05/2020	**Complert**
Desenvolupament del *Service Worker* (PWA)	08/05/2020	08/05/2020	**Complert**
Desenvolupament i execució de proves (JEST)	09/05/2020	13/05/2020	**Complert**

8. Demostració

A continuació es detallen les passes a seguir per tal d'executar l'aplicació en local a qualsevol màquina.

8.1. Requisits

Cal tenir instal·lades les següents tecnologies:

Node.js	versió 11.14.0 o superior
NPM	versió 6.7.0 o superior
mongoDB	versió 3.4.2 o superior servidor a localhost:27017 (defecte)

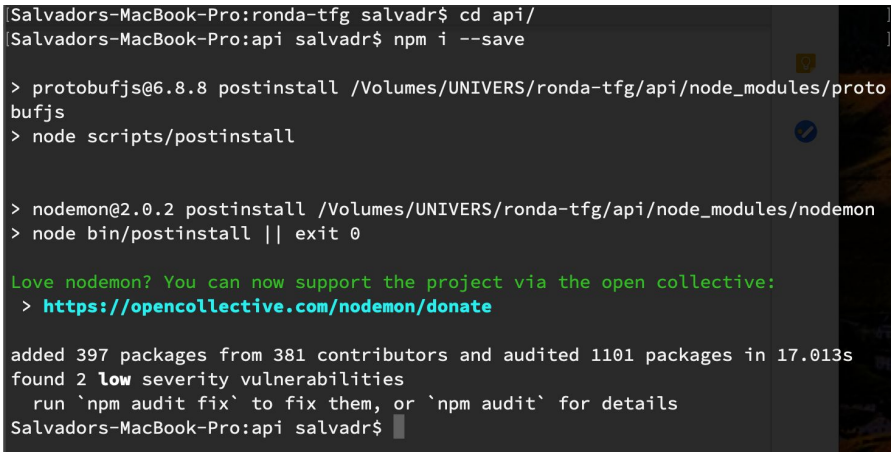
8.2. Instal·lació

Un cop es descomprimeix el codi font hauria de quedar la següent estructura:

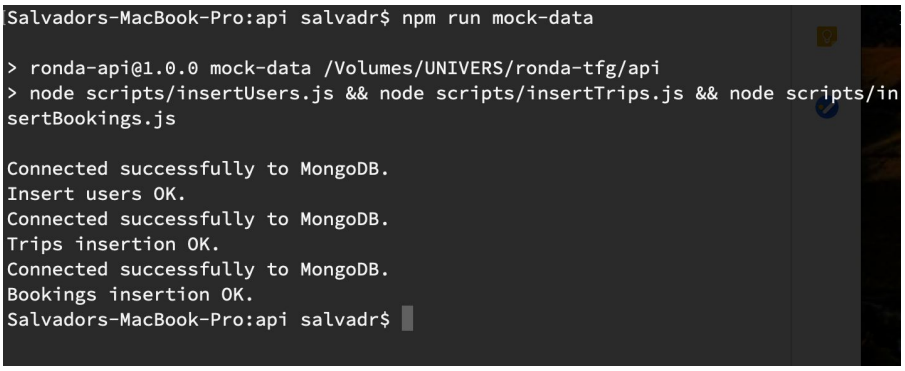
```
../
     /api
     /app
```

A continuació cal instal·lar les dependències per a les dues aplicacions.

Dins `/api`, executem **`npm i --save`**:

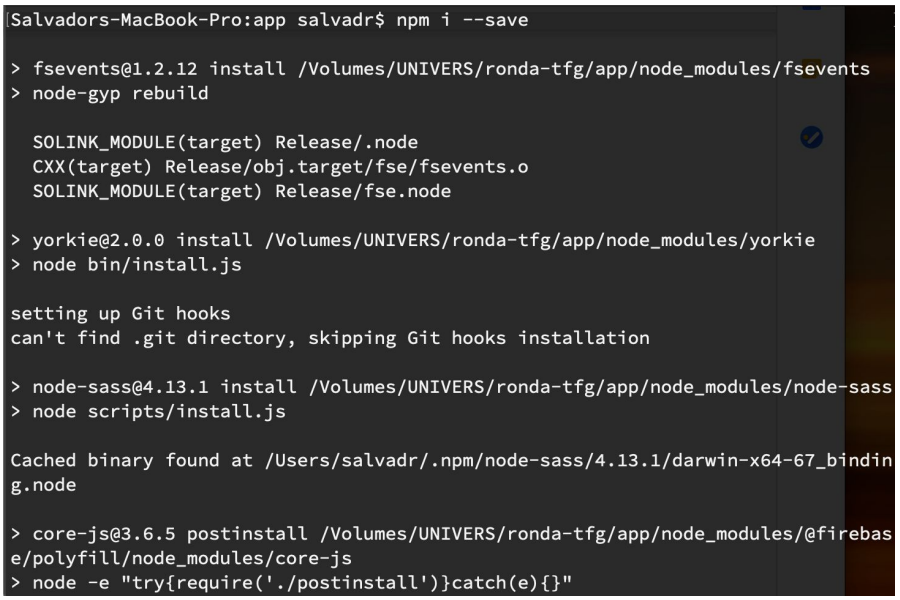


S'ha inclòs una comanda per instal·lar dades de prova per tal de facilitar la revisió, executem **npm run mock-data:**

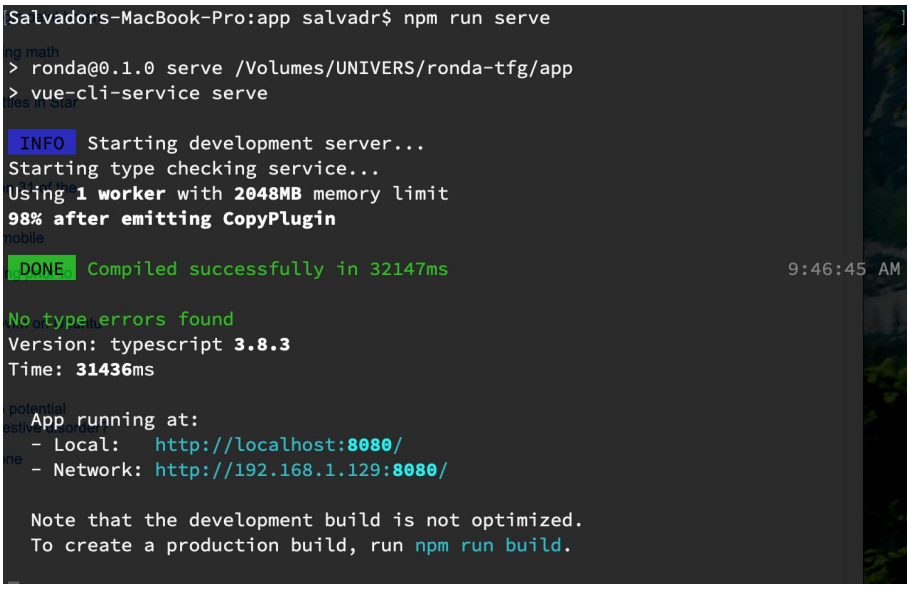


Ja podem executar l'API amb **npm run watch:**

A continuació instal·lem les dependències dins `/app`:



Ara ja és possible executar l'aplicació (*front-end*) amb **`npm run serve`**:

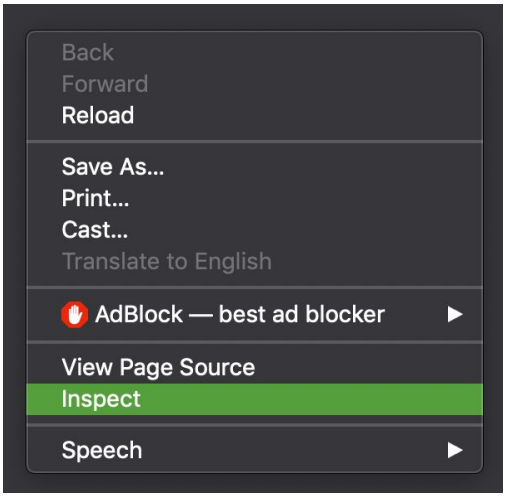


8.3. Accés

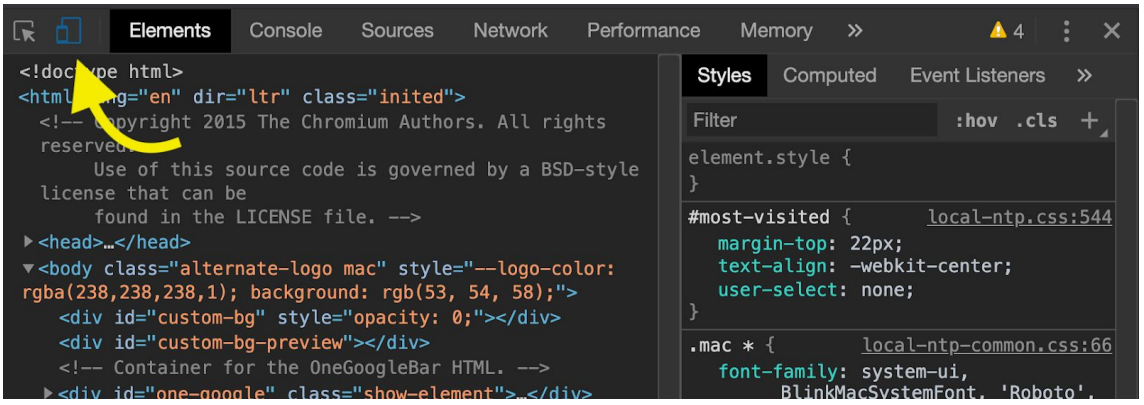
Per defecte, si els ports es troben lliures, l'adreça per accedir a l'aplicació és **localhost:8080.**

A continuació es detallen les passes per tenir utilitzar l'aplicació amb una experiència òptima:

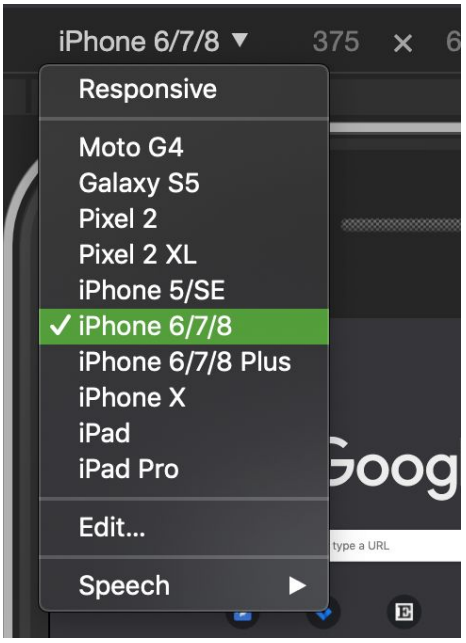
- Accedim a l'inspector del navegador fent clic al botó dret dins de la pàgina i després a *Inspect* (les imatges pertanyen al navegador Chrome):



Dins l'inspector, fem clic al botó indicat per veure la pàgina amb resolució de dispositiu mòbil:

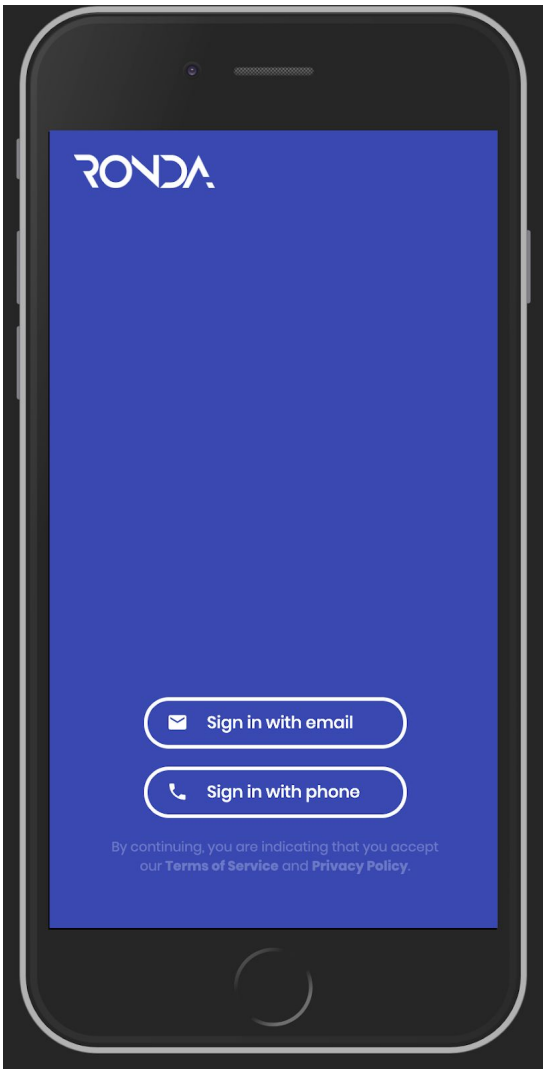


Seleccionem *iPhone 6/7/8* a la pestanya de dispositius (a la part superior de la pantalla principal):



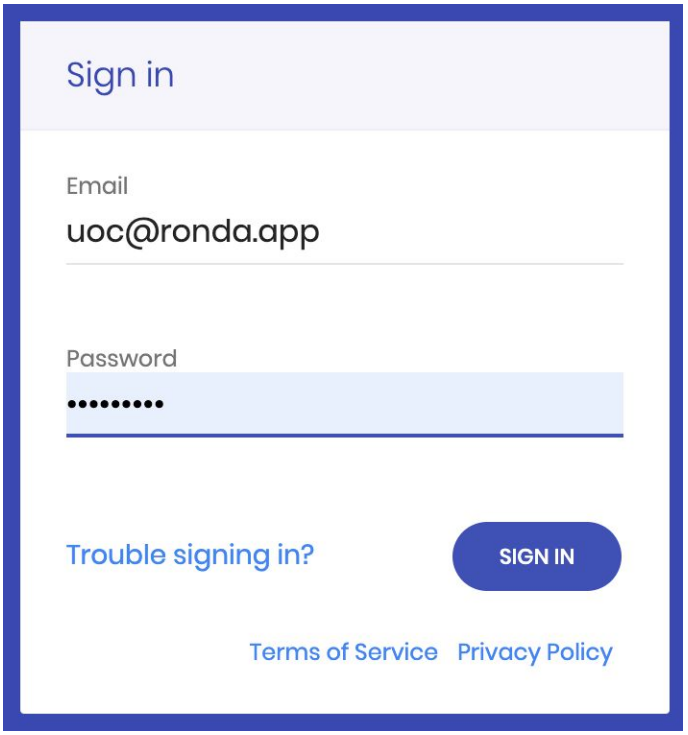
Per més realisme es pot afegir el marc del dispositiu al menú de la part superior dreta:

Accedim a **localhost:8080**:



Per accedir, cal triar l'opció amb correu electrònic i utilitzar els següents credencials:

E-mail: **uoc@ronda.app**
Clau: **123123123**



Un cop dins s'ha de completar el procés d'introducció (totes les dades ja estan guardades prèviament, només cal avançar) s'arriba a l'agenda, on s'haurien de veure diverses reserves, les quals poden servir per veure com seria l'aplicació en un entorn real.

9. Conclusions

Un cop finalitzat el treball final de grau es pot concloure que s'han complert els objectius plantejats a l'inici d'aquest:

- S'ha dissenyat un producte des de zero, específic per *car pooling* laboral, el qual permetrà aprofitar milers de places de cotxe buides que viatgen cada dia a la capital catalana.
- El producte s'ha desenvolupat utilitzant tecnologies web d'última generació dins el paradigma d'aplicació progressiva (PWA), així com una API que permet l'autenticació, la persistència de les dades i les notificacions en temps real.
- L'aplicació reflecteix l'estat de l'art actual envers el desenvolupament d'aplicacions per dispositius mòbils híbrides, a través de l'ús de *service workers* i serveis com *Firebase*.

Durant el disseny he trobat algunes dificultats amb aquells components innovadors, creats específicament per aquesta aplicació. En diverses ocasions he hagut de realitzar iteracions improductives, però que m'han conduït al resultat final.

Respecte al desenvolupament, la familiaritat prèvia amb el nucli de tecnologies utilitzades ha facilitat el procés i he pogut dedicar més temps a aquelles tecnologies que he utilitzat per primera vegada, com els *service workers* i els *websockets*. En aquests casos, atès que es tracta de tecnologies relativament recents, he trobat dificultats amb la documentació, ja que la ràpida evolució de les mateixes fa que quedi obsoleta i sense consistència. Tot i això, finalment he pogut desenvolupar les funcionalitats necessàries.

Tot i haver assolit els objectius plantejats, hi ha diverses tasques que resultarien necessàries abans de fer pública l'aplicació, tals com:

- Implementar un sistema de comunicació entre usuaris.
- Connectar amb una plataforma de pagament per realitzar pagaments amb targeta des de l'aplicació.
- Permetre el seguiment en temps real del vehicle.

En definitiva, l'aplicació desenvolupada es podria considerar un PMV (producte mínim viable) el qual servirà per presentar el concepte i realitzar proves amb usuaris, i un cop es consideri que existeix una validació, continuar el desenvolupament fins a arribar al producte final.

10. Glossari

API
Application Programming Interface (Interfície de Programació d'Aplicacions)

Backend
Aplicatiu que s'executa a un servidor i es comunica amb el *frontend* (abaix).

Car pooling
L'activitat de compartir un vehicle entre diversos usuaris, quan un d'ells n'és el propietari i conductor.

Commuting
Desplaçaments d'anada i tornada, normalment aquells relacionats amb el lloc de treball.

Dating
Cites entre persones que busquen parella.

DOM
Document Object Model, model d'objecte del document.

Networking
Activitat de connectar amb d'altres persones dinsun àmbit professional.

Framework
Aplicatiu que reuneix funcions genèriques pel desenvolupament d'aplicatius específics.

Frontend
Aplicatiu que s'executa en un client (per exemple, a un navegador) i es comunica amb el *backend*.

GIT
Aplicatiu de control de versions de codi font.

HTTP(S)
Hyper-text Transfer Protocol (Secure), protocol de transmissió d'híper text (segur).

JSON
JavaScript Object Notation, notació d'objectes JavaScript.

JWT
JSON Web Token, mètode criptogràfic per assegurar comunicacions en format JSON.

Middleware
Funció dins un aplicatiu de servidor que s'executa entre la crida i la resposta.

MVVM
Patró de disseny d'aplicatius en relació amb el desenvolupament d'interfícies gràfiques.

Offline
Que no té connexió a Internet.

PWA
Progressive Web Application, aplicació web progressiva.

Socket (network)
Extrem que connecta amb un altre extrem dins una xarxa i permet la comunicació entre dos nodes.

SPA
"Single Page Application" (Aplicació d'una sola pàgina). Un patró de desenvolupament d'aplicacions web que consisteix en l'ús de crides asíncrones, és a dir, el navegador web només carrega l'aplicació una única vegada (a l'inici).

SSL
Secure Sockets Layer, capa segura de comunicació a través de *sockets*.

Stack
Pila.

Token
Símbol, identificador, peça.

11. Bibliografia

[1] Environmental impact of transport, Wikipedia, https://en.wikipedia.org/wiki/Environmental_impact_of_transport (2 de març de 2020)

[2] La vivienda sube un 48,8% en Barcelona en cinco años, La Vanguardia, https://www.lavanguardia.com/economia/20180809/451260870681/pisos-barcelona-precios-subida.html (2 de març de 2020)

[3] Sant Cugat, Sabadell y Terrassa, los municipios del área metropolitana que ganan más población, La Vanguardia, https://www.lavanguardia.com/local/20120920/54350821942/sant-cugat-sabadell-terrassa-mas-habitantes-ganan.html (3 de març de 2020)

[4] Atascos: 52.000 horas perdidas cada día, RACC, http://blog.racc.es/coche/atascos-horas-perdidas/ (2 de març de 2020)

[5] Así ha cambiado la población en España desde 2009, por municipios y distritos, El País, https://elpais.com/sociedad/2020/01/22/actualidad/1579682422_850330.html (19 de maig de 2020)

[6] Estudi del vehicle compartit a Barcelona: anàlisi, Ajuntament de Barcelona, **https://bcnroc.ajuntament.barcelona.cat/jspui/bitstream/11703/106643/1/Estudi%20del%20vehicle%20compartit%20a%20Barcelona_An%c3%a0lisi_Octubre2017.pdf** (19 de maig de 2020)

[7] Evolució dels preus de l'Autoritat del Transport Metropolità, Wikipedia, https://ca.wikipedia.org/wiki/Evoluci%C3%B3_dels_preus_de_l%27Autoritat_del_Transport_Metropolit%C3%A0 (18 de març de 2020)

[8] The Progressive Javascript Framework, Vue.js, https://vuejs.org/ (20 d'abril de 2020)

[9] What is Vuex?, Vue.js Vuex, https://vuex.vuejs.org/ (20 d'abril de 2020)

[10] Service Worker | An Introduction, Web Fundamentals (Google), https://developers.google.com/web/fundamentals/primers/service-workers (20 d'abril de 2020)

[11] Firebase, firebase, https://firebase.google.com/ (20 d'abril de 2020)

[12] node.js, node.js, https://nodejs.org (20 d'abril de 2020)

[13] mongoDB, mongoDB, https://www.mongodb.com/ (20 d'abril de 2020)

[14] Architecture of Node.js' Internal Codebase, Medium, http://web.archive.org/web/20170208132221/https://arenli.com/architecture-of-node-js-internal-codebase-57cd8376b71f?gi=1984716e61d6 (20 d'abril de 2020)

[15] Firebase UI, Github, https://github.com/firebase/firebaseui-web (21 d'abril de 2020)

[16] JSON Web Tokens, JWT, https://jwt.io/ (21 d'abril de 2020)

[17] Koa - next generation web framework for node.js, Koa, https://koajs.com/ (22 d'abril de 2020)

[18] Jest, Jest, https://jestjs.io/ (1 de maig de 2020)